RSAC 2025 Conference

Many Voices. One Community.

SESSION ID: CLS-R01

Beauty and the ALBeast: Be our (Shared Responsibility Model) Guest

Liad Eliyahu
Head of Research
Miggo Security

Itai Goldman
Co-founder & CTO
Miggo Security

It’s time to meet the Beast

- 15,000 potentially vulnerable instances (F500, Governments)
- Acknowledgement for multiple CVEs in OSS projects

CVE-2024-10125 **CVE-2024-8901**	aws
CVE-2025-1293	HashiCorp
CVE-2025-25182	GCHQ

It’s time to meet the Beast

- Sheds light on a broader issue within the ***Shared Responsibility Model***

Miggo disclaimer

This recorded presentation is provided by Miggo Security, Inc. for informational and educational purposes only. The content presented reflects the views and opinions of the presenters and does not necessarily represent the official position of Miggo Security, Inc.Viewers should exercise independent professional judgment when applying any information presented. Miggo Security, Inc. makes no warranties, express or implied, regarding the accuracy, completeness, or reliability of the content and assumes no liability for any actions taken based on the information provided.

This recording is the property of Miggo Security, Inc. and is protected by copyright laws. Unauthorized reproduction, distribution, or use of this material in whole or in part is strictly prohibited. By viewing this presentation, you acknowledge and agree to these terms and conditions

Miggo and RSAC Conference logos and other trademarks are proprietary. All rights reserved.

In today's session we'll:

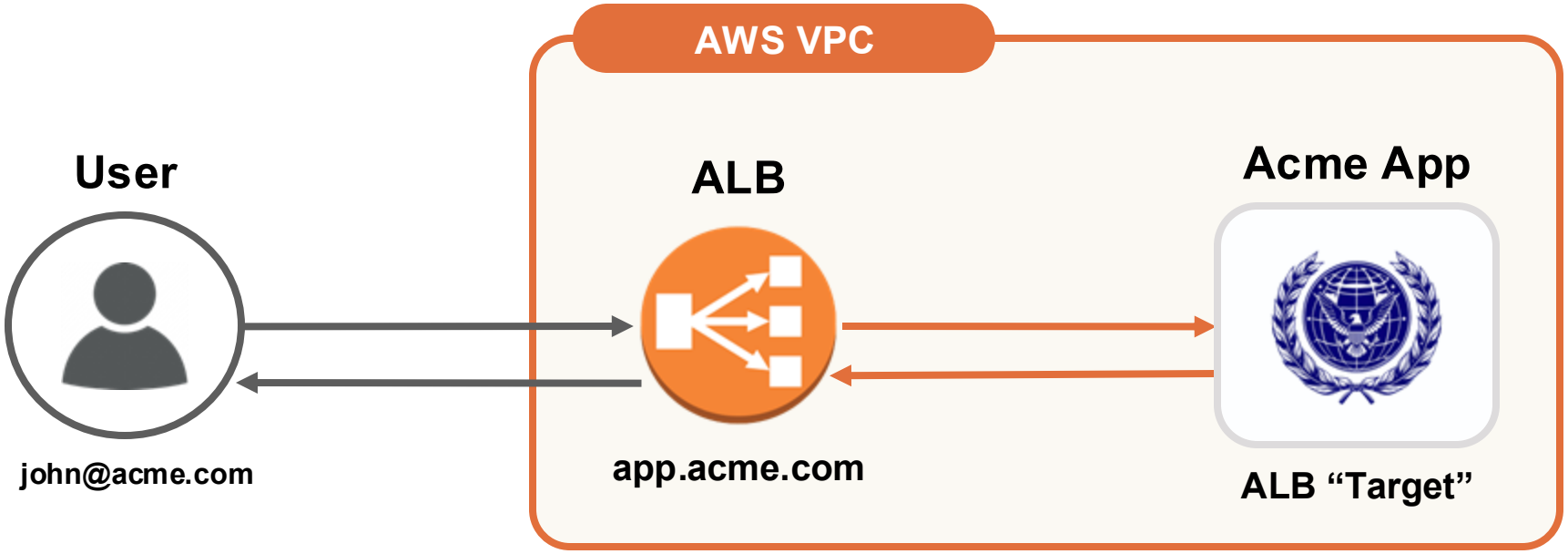
1. Understand the ALBeast issue
2. Share the lessons learned about the broader problem
3. Offer practical recommendations

$ whoami

Liad Eliyahu
Head of Research
Miggo Security

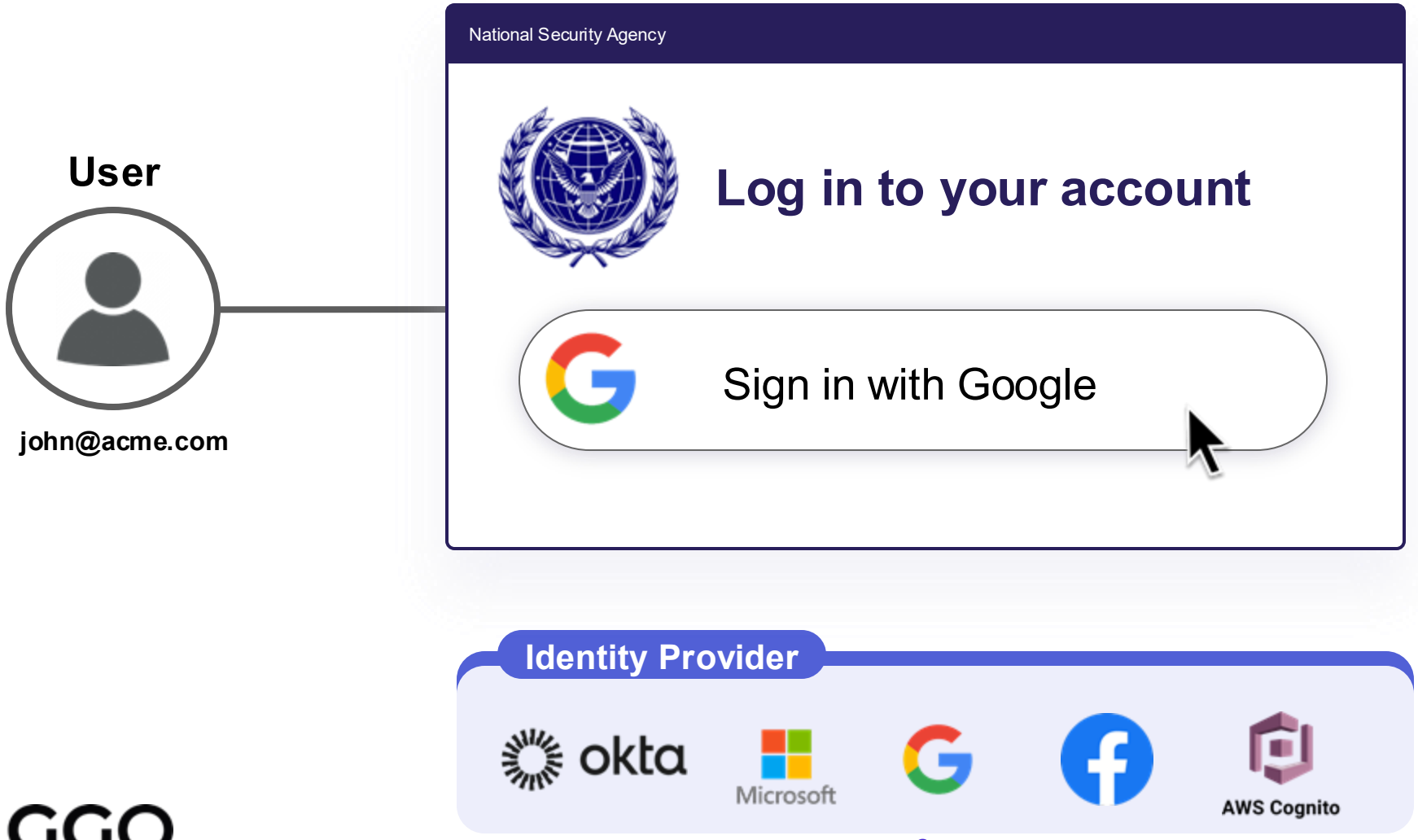
Itai Goldman
Co-founder & CTO
Miggo Security

How does an ALB work?



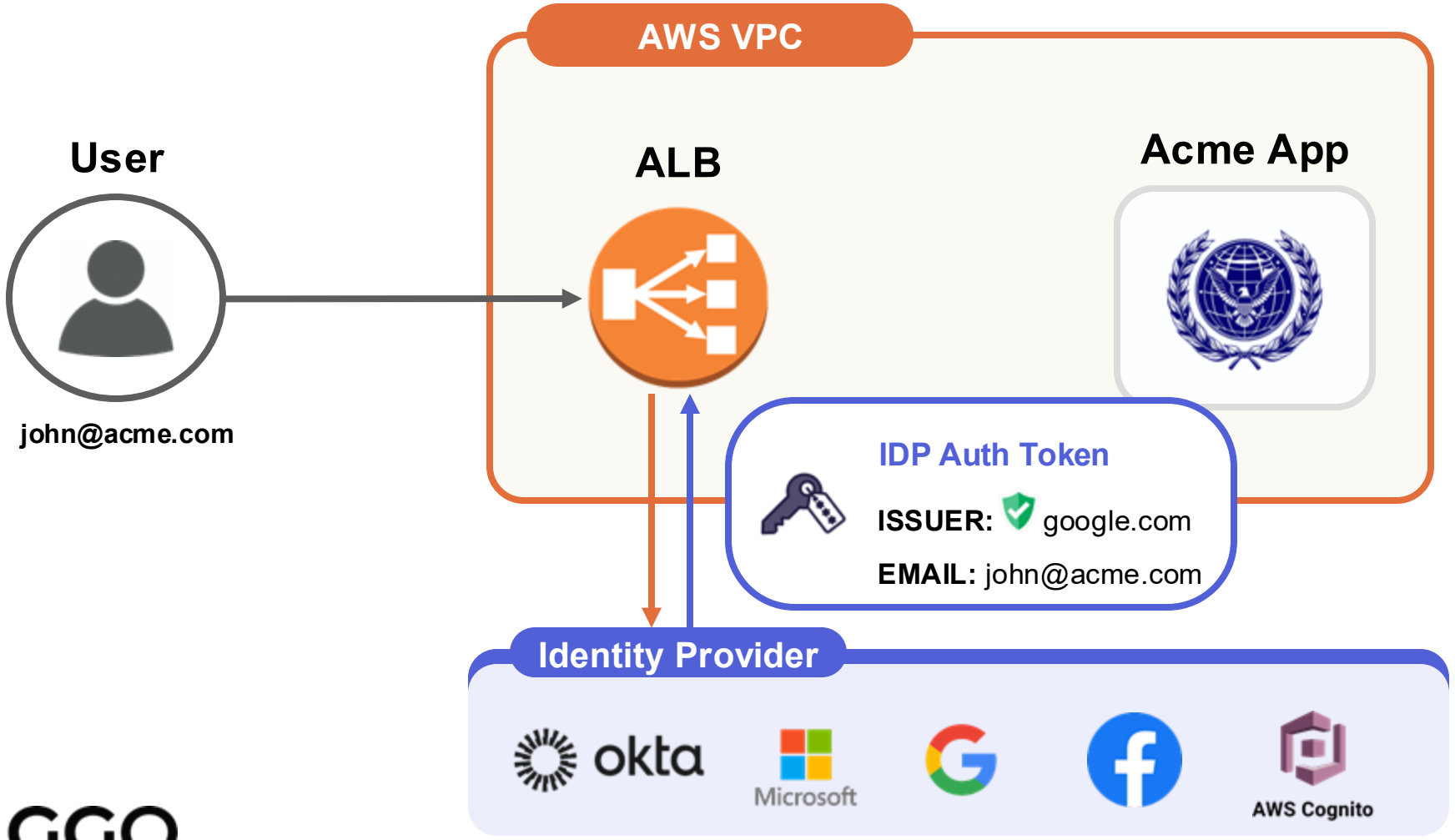
What is ALB Authentication?

User sends request to the application



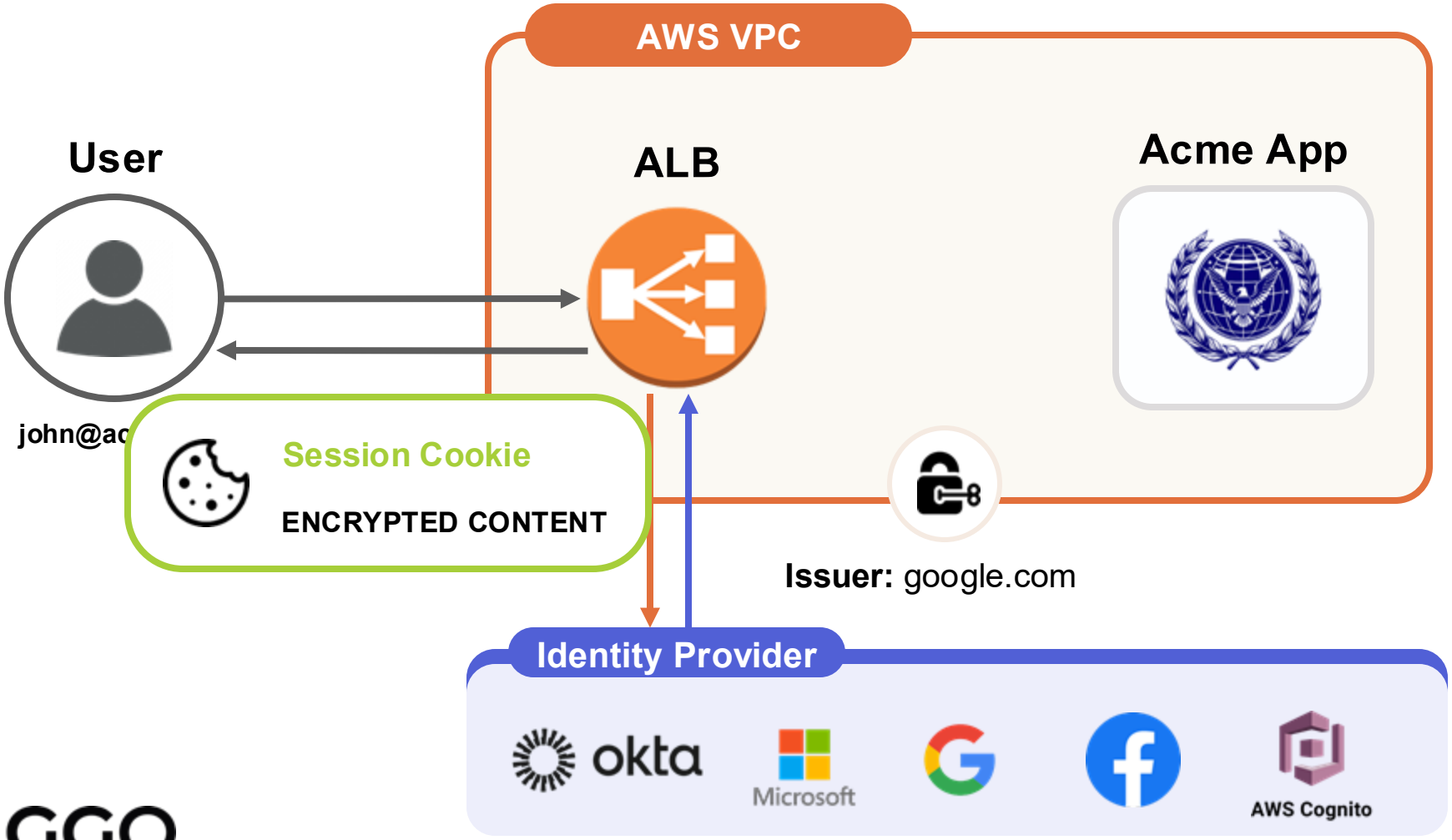
What is ALB Authentication?

The ALB gets the claims token from configured IDP



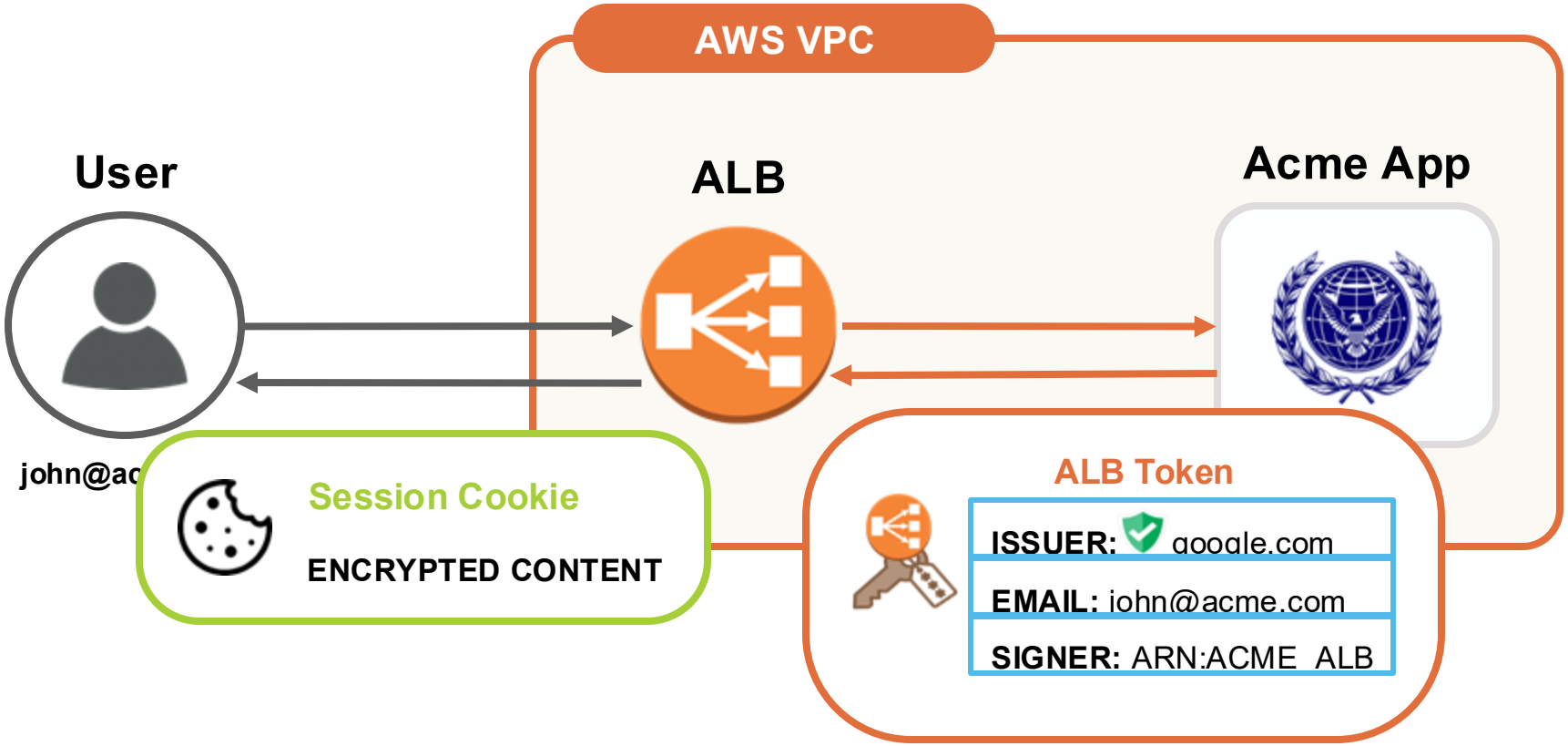
What is ALB Authentication?

ALB creates an encrypted identification cookie for the user

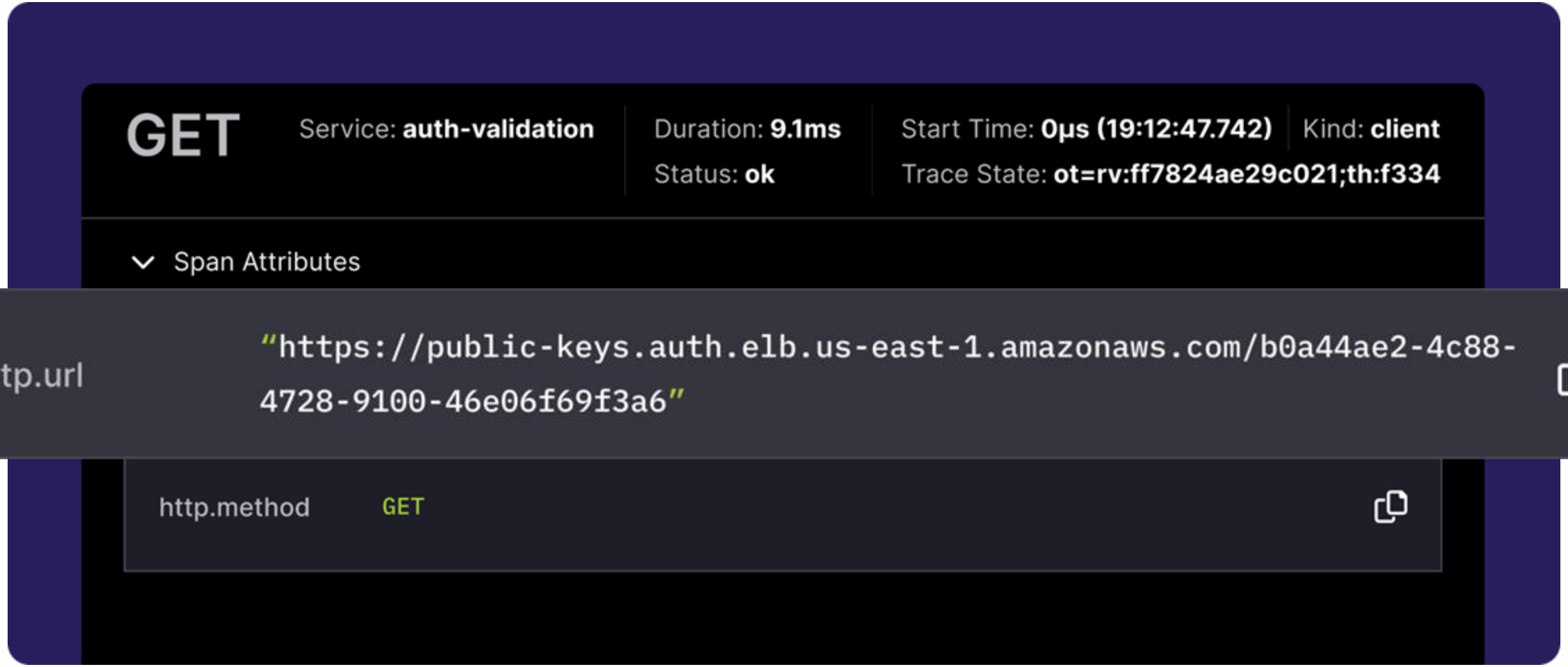


What is ALB Authentication?

ALB signs a new token with claims and passes it for the application



How did we stumble across the beast?



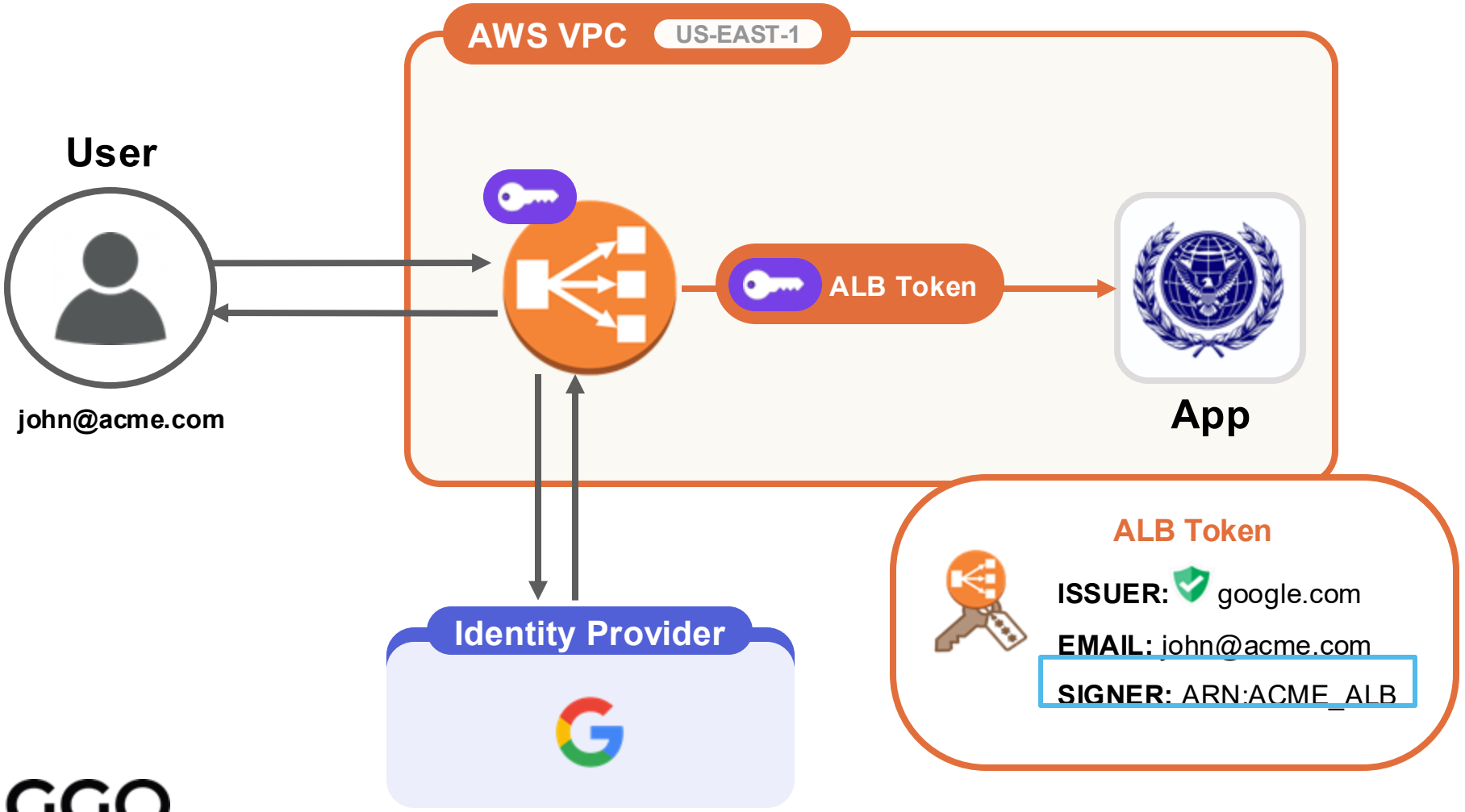
What was the issue?

- Every ALB instance can sign a token that could be validated by any other ALB instance in the same region

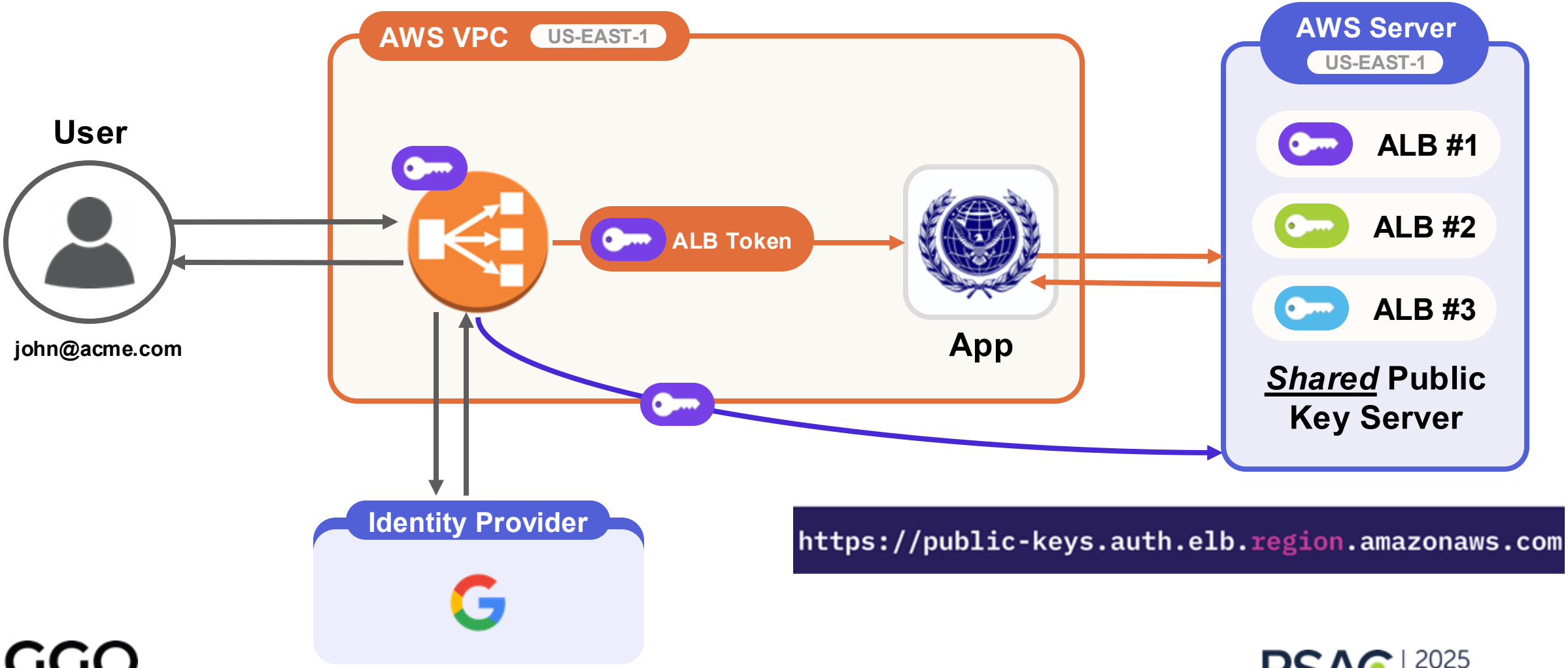
```
# Step 2: Get the public key from regional endpoint
url = 'https://public-keys.auth.elb.' + region + amazonaws.com/' + kid
```

- Instance = application **!**

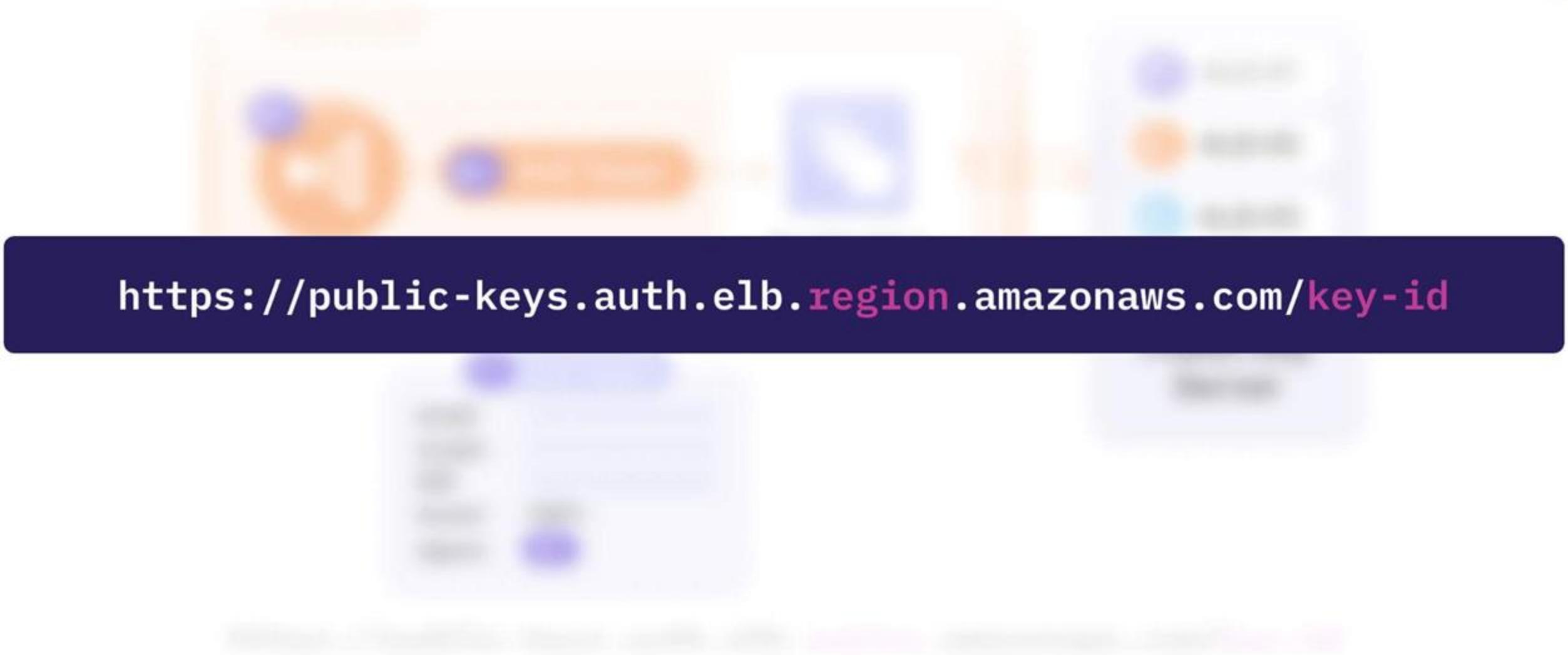
How a token is signed in the region



How a token is signed in the region



How a token is validated in the region



Where is the signer?

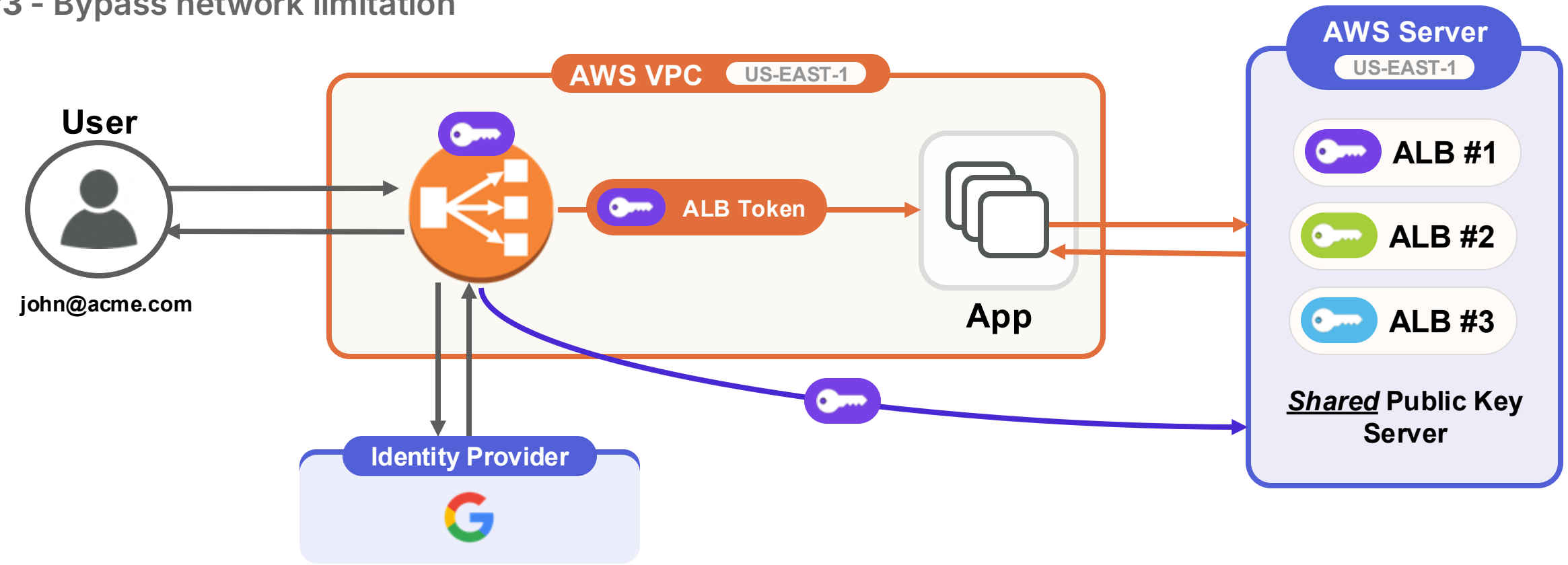
```python
# Step 1: Get the key id from JWT headers (the kid field)
encoded_jwt = headers.dict['x-amzn-oidc-data']
jwt_headers = encoded_jwt.split('.')[0]
decoded_jwt_headers = base64.b64decode(jwt_headers)
decoded_json = json.loads(decoded_jwt_headers)
kid = decoded_json['kid']

# Step 2: Get the public key from regional endpoint
url = 'https://public-keys.auth.elb.' + region + amazonaws.com/' + kid
req = requests.get(url)
pub_key = req.text
```

Three challenges to overcome

#1 - Bypass signer validation

#2 - Bypass issuer validation

#3 - Bypass network limitation

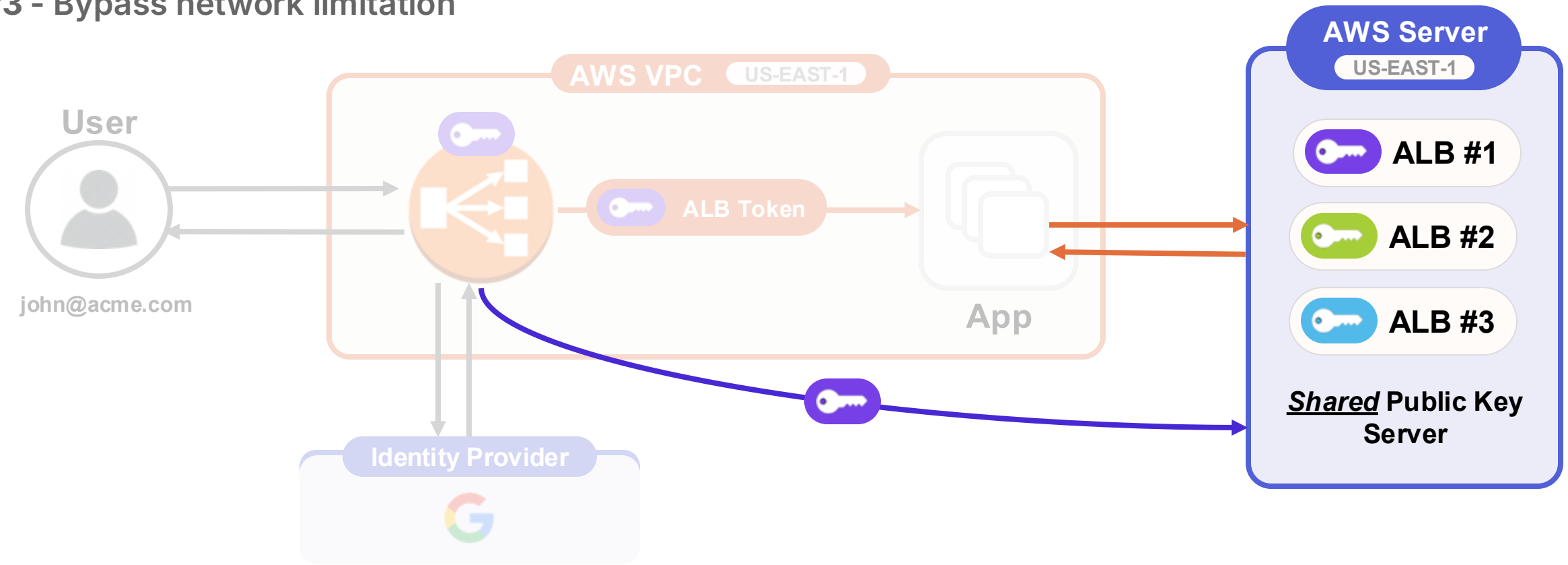


Three challenges to overcome

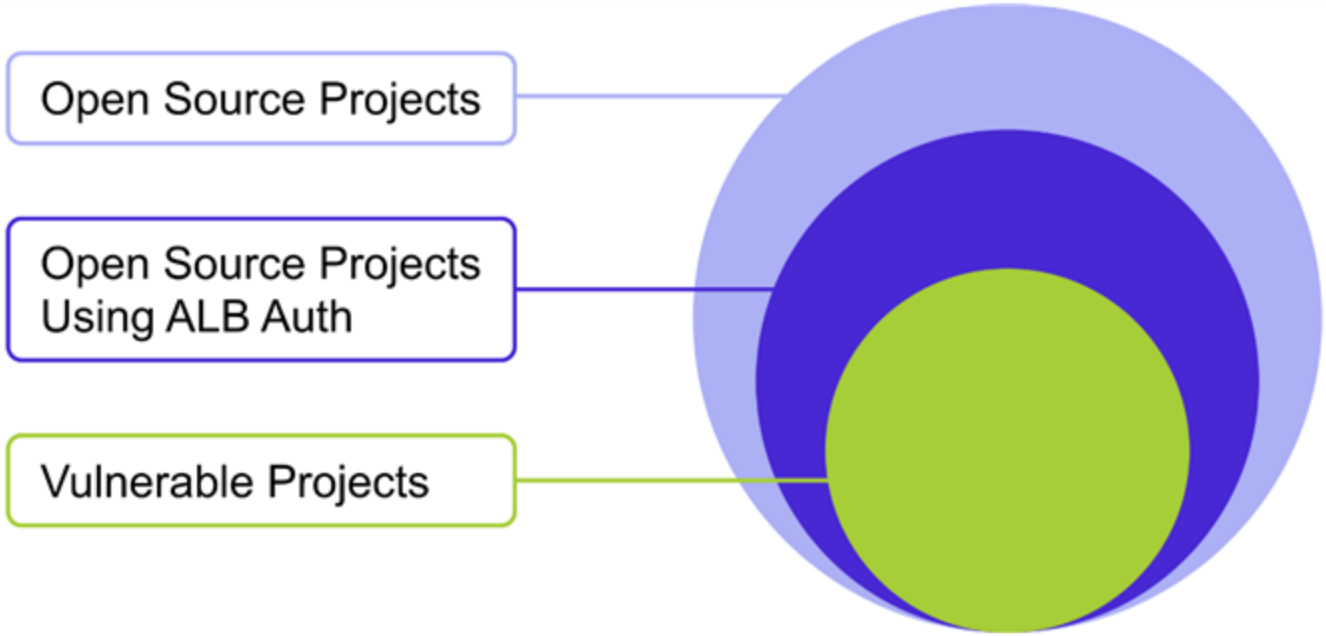
#1 - Bypass signer validation ✅

#2 - Bypass issuer validation

#3 - Bypass network limitation



Investigating the blast radius



https://p

key-id

Investigating the blast radius

aws

GCHQ

NETFLIX

HashiCorp

Spotify®

The Guardian

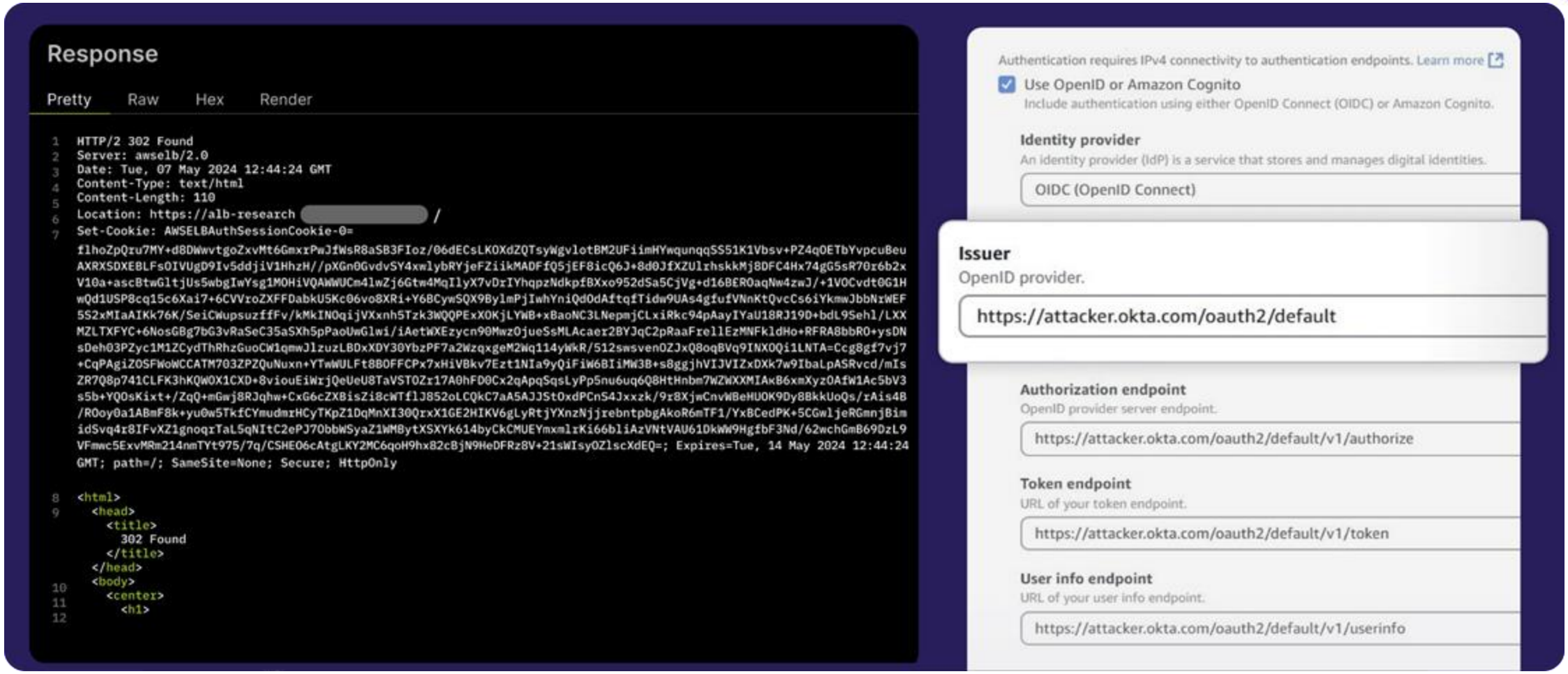
Who's using ALB authentication safely?

	Validate the signer	Validate the issuer
66%	✗	✗
34%	✗	✓

Normally the issuer validation would reject an attacker

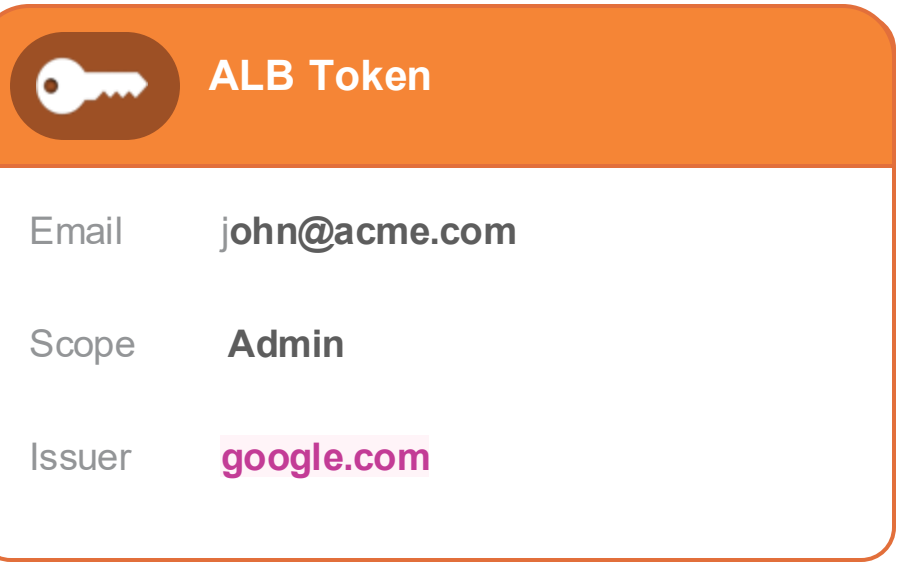
What if an attacker could control the issuer field in the token?

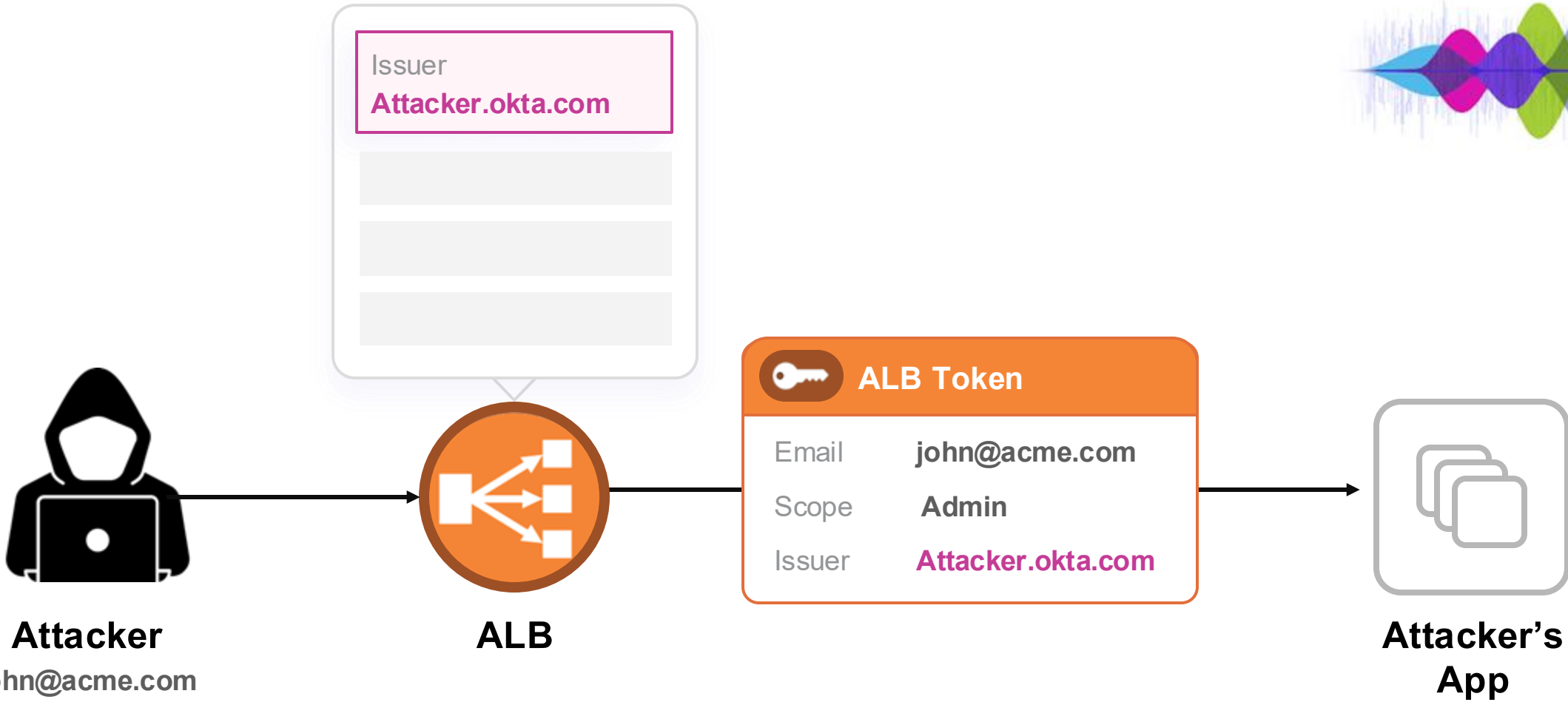
Where the issuer lives: cookies and configuration

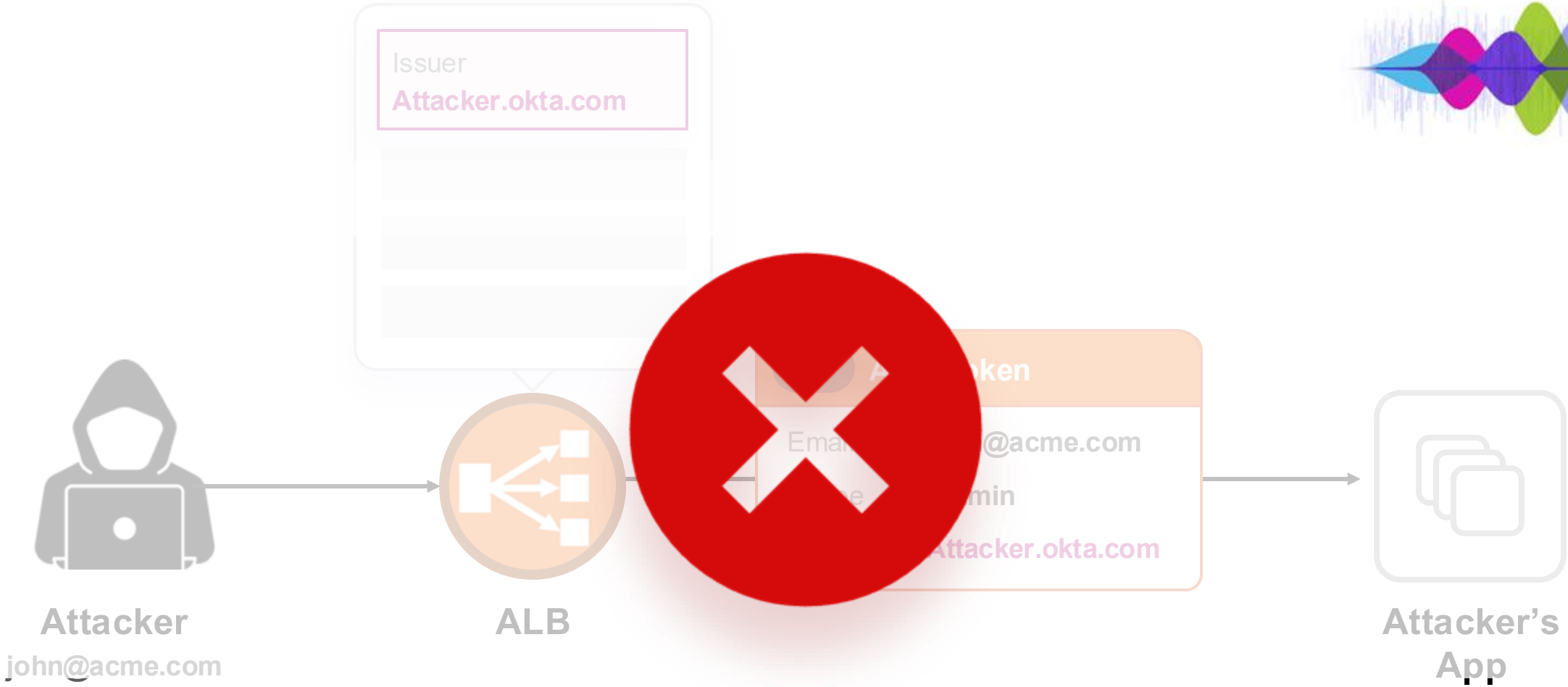


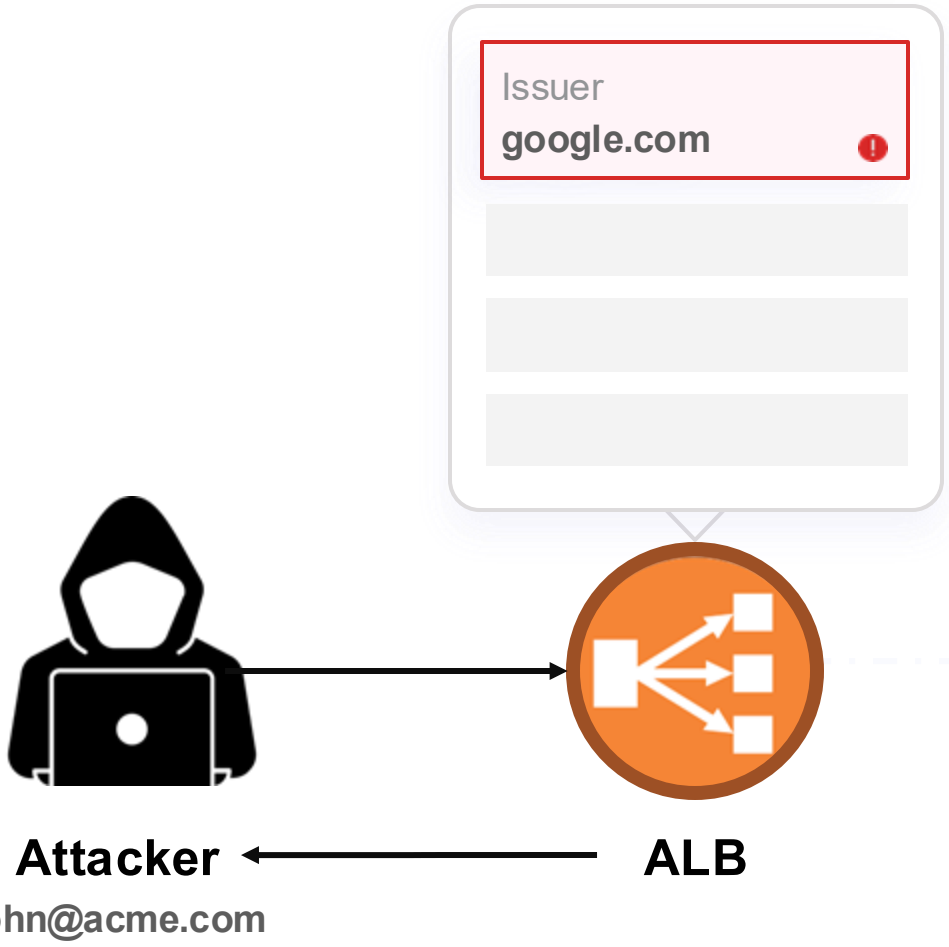
Configuration wins!

Gaining control over the issuer




Issuer
Attacker.okta.com
ALB Token
Email john@acme.com
Scope Admin
Issuer Attacker.okta.com
Attacker
john@acme.com
ALB
Attacker's App

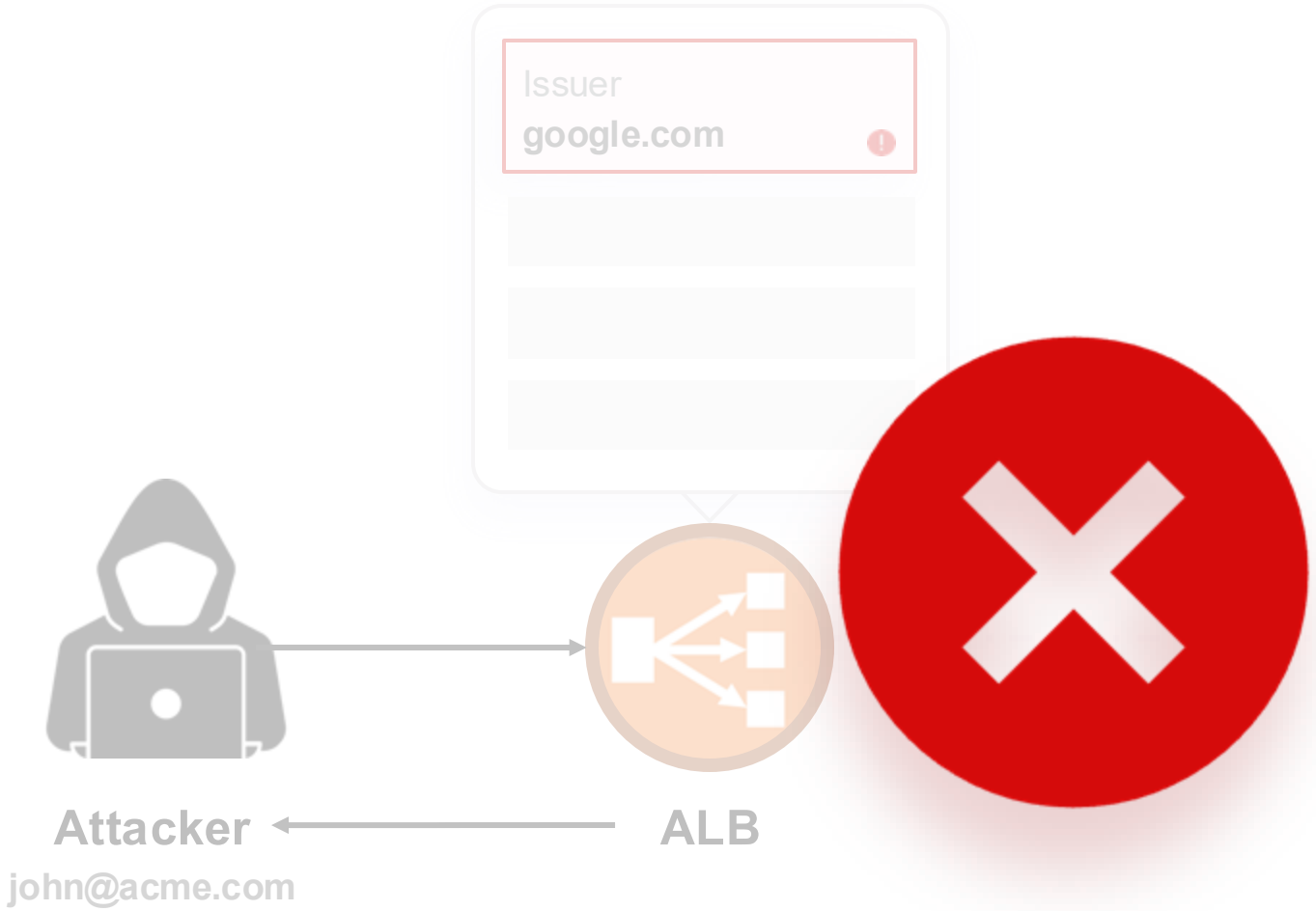

Issuer
Attacker.okta.com
@acme.com
Attacker.okta.com
Attacker
john@acme.com
ALB
Attacker's App

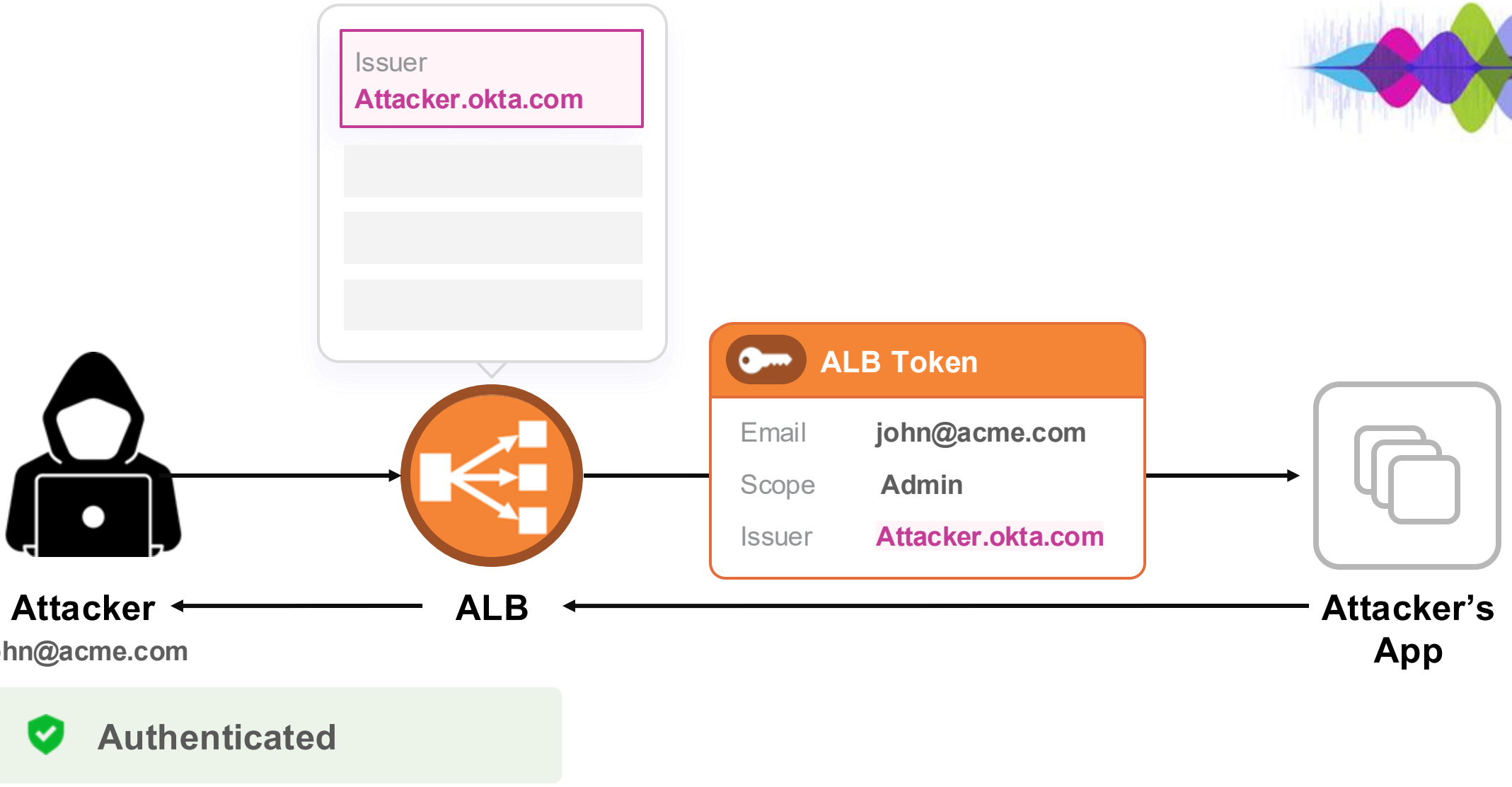


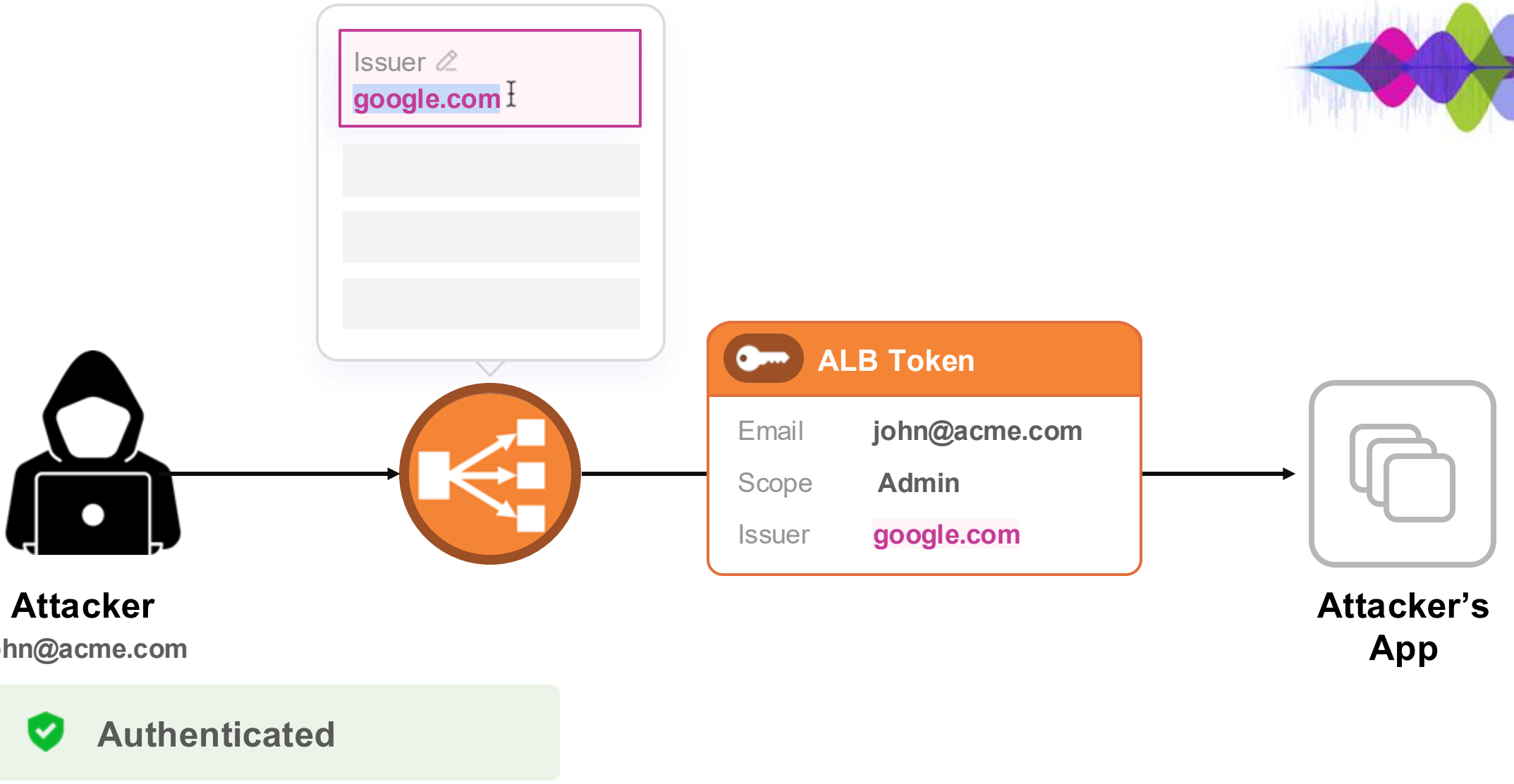
Attacker's App

Authentication failure

You are not google


Issuer
google.com
Attacker
john@acme.com
ALB
Attacker's App
Authentication failure
You are not google


Issuer
Attacker.okta.com
ALB Token
Email john@acme.com
Scope Admin
Issuer Attacker.okta.com
Attacker
john@acme.com
ALB
Attacker's App
Authenticated


Issuer
google.com
ALB Token
Email john@acme.com
Scope Admin
Issuer google.com
Attacker
john@acme.com
Attacker's App
Authenticated

Issuer
google.com

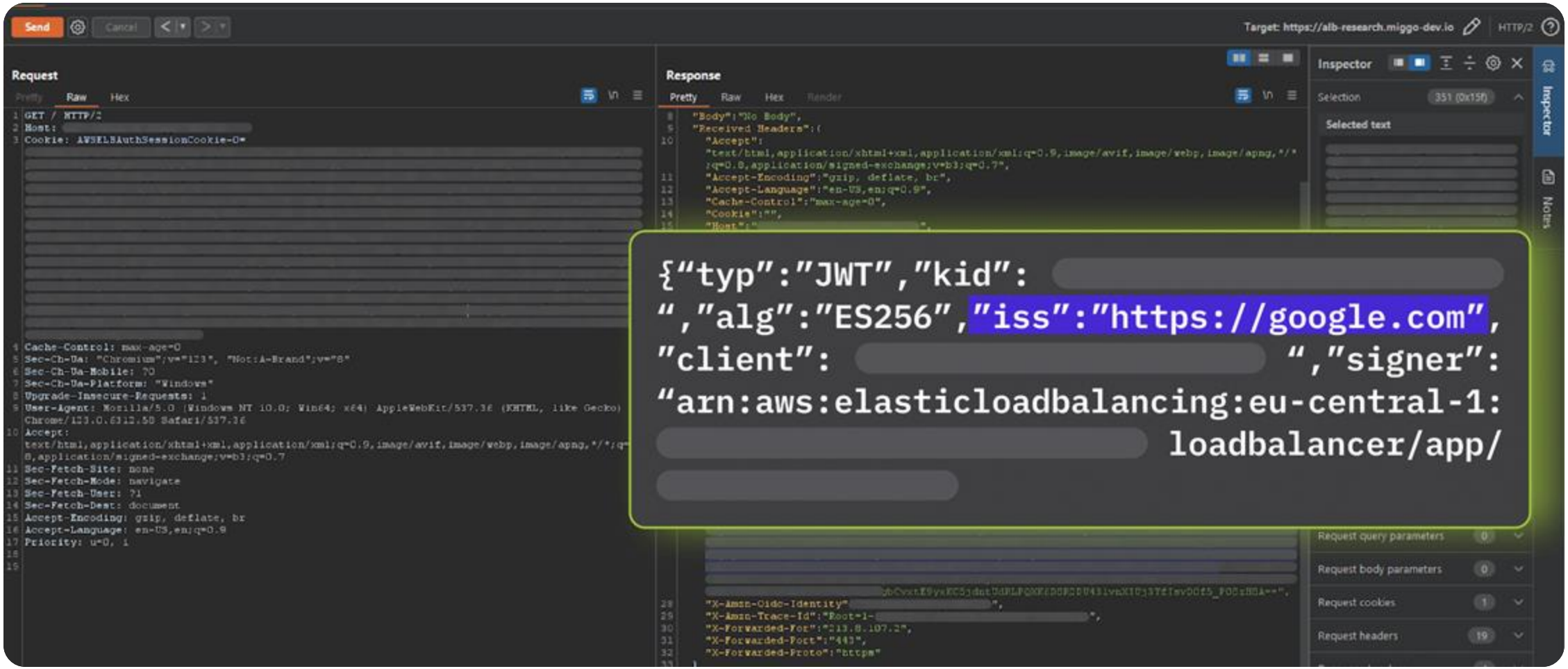
Attacker
john@acme.com

Attacker's App

Authenticated

Suc rging

The forged issuer: I'm Google now



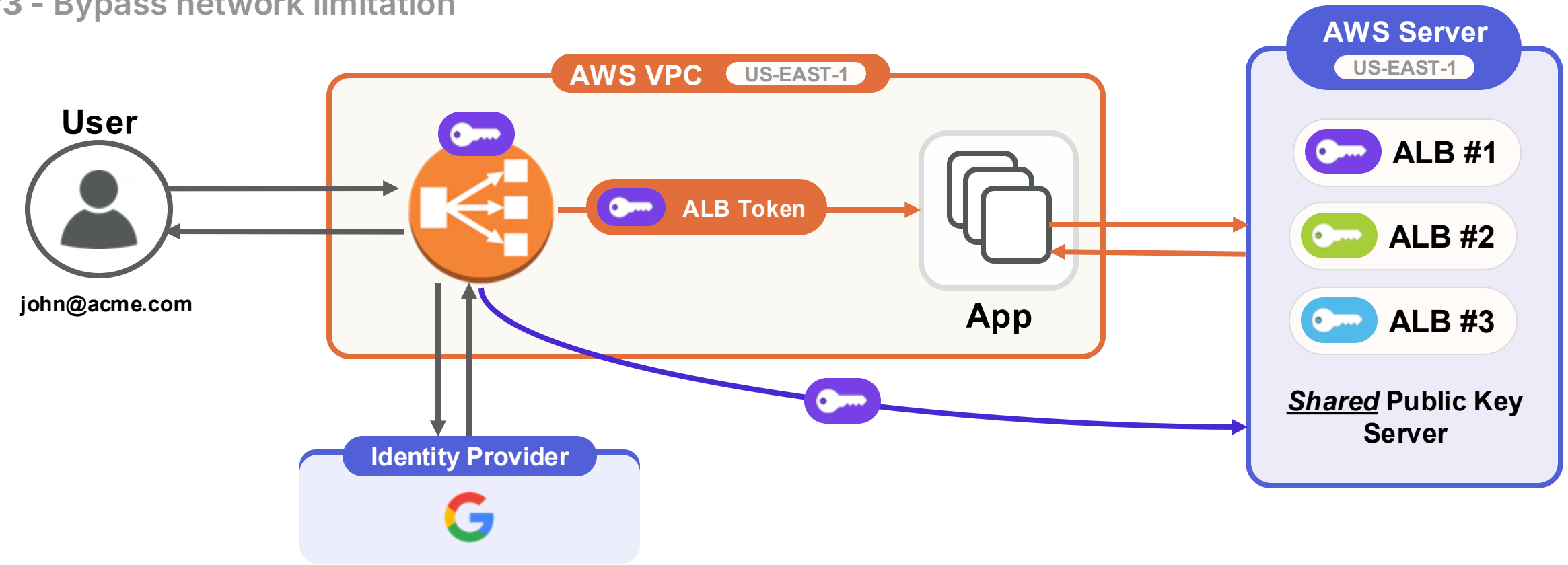
Why was it possible?

Three challenges to overcome

#1 - Bypass signer validation ✅

#2 - Bypass issuer validation

#3 - Bypass network limitation

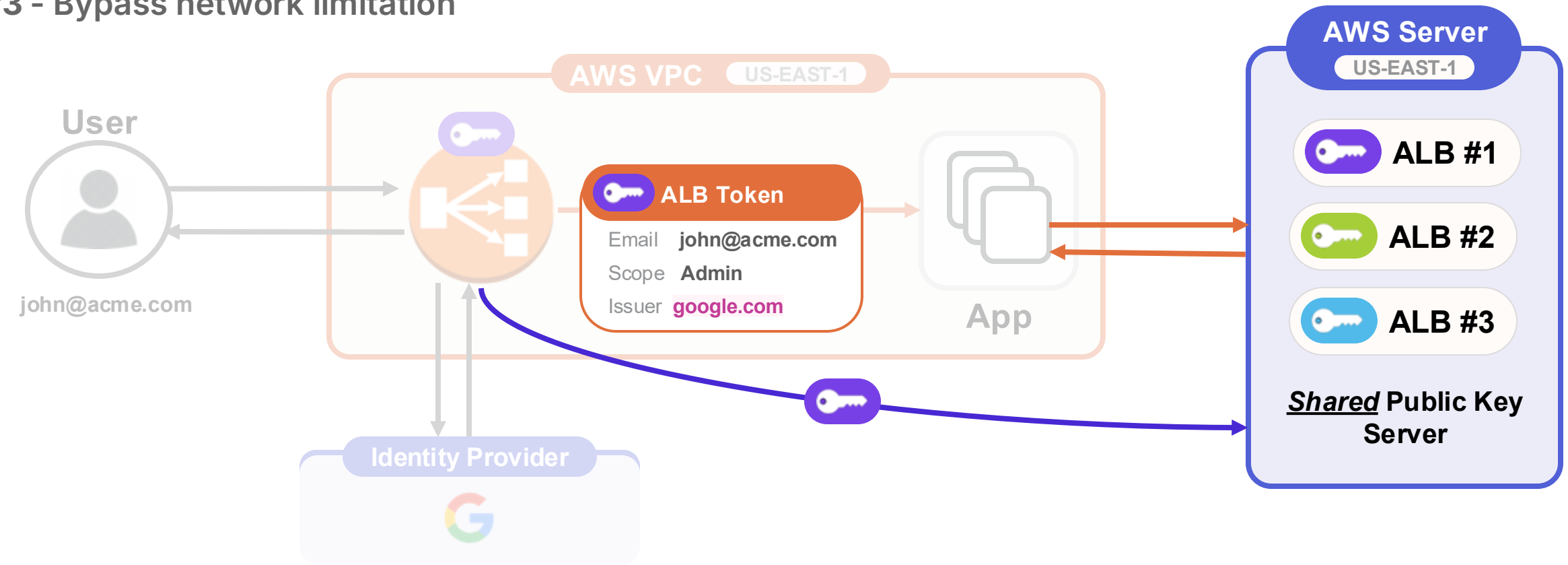


Three challenges to overcome

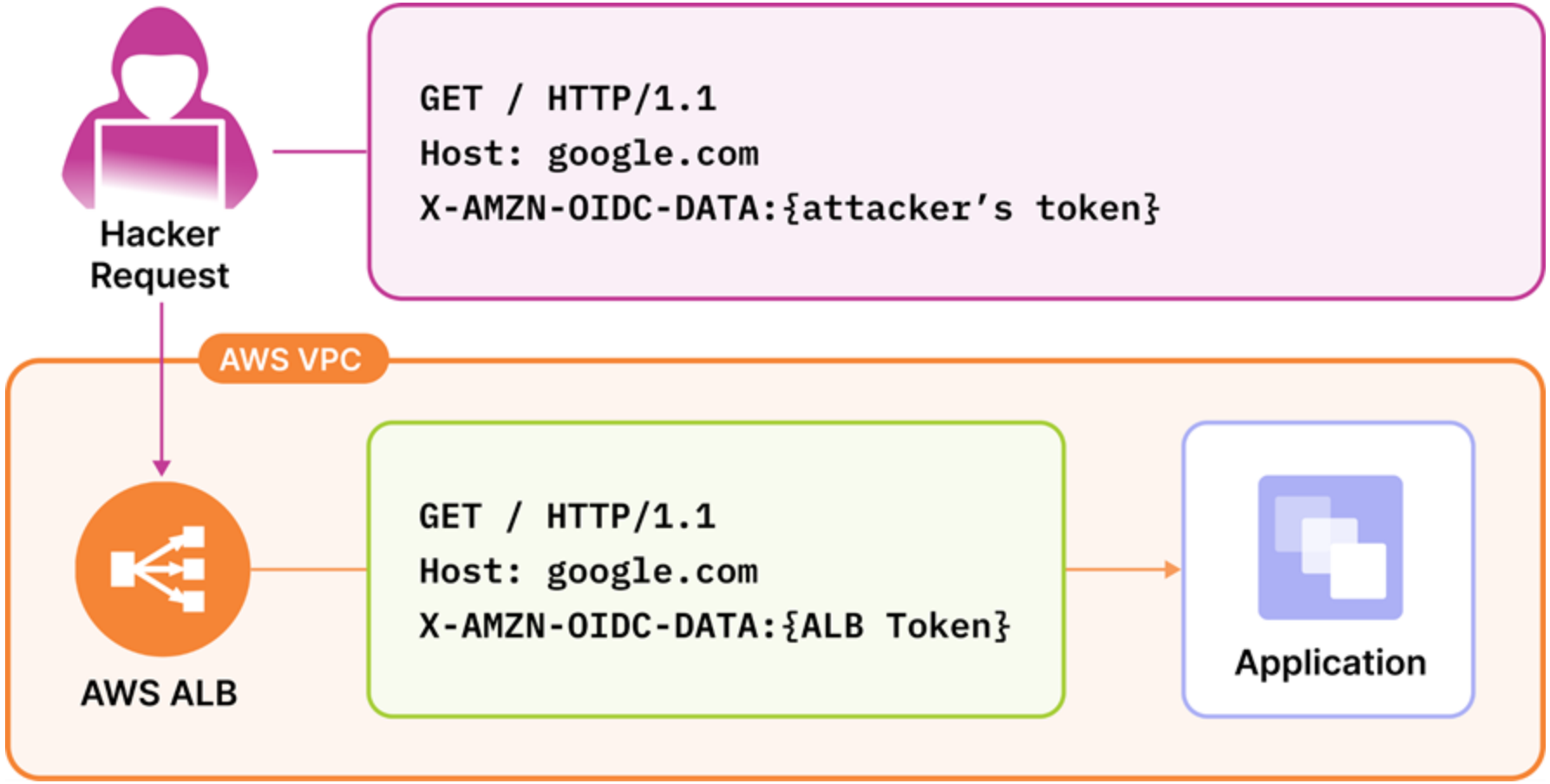
#1 - Bypass signer validation ✅

#2 - Bypass issuer validation ✅

#3 - Bypass network limitation



2 down, 1 to go - for the attacker



Bypassing the ALB network

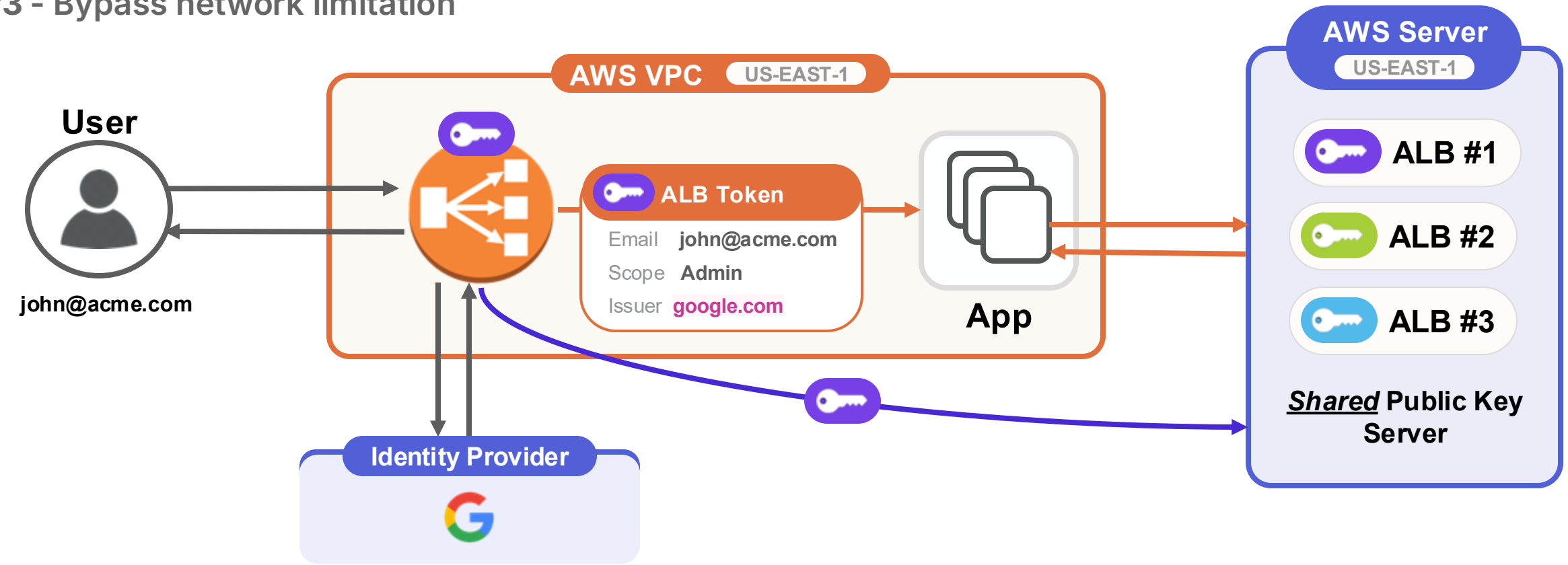
- Bypas ps”
- Applic
- Only tion

Three challenges to overcome

#1 - Bypass signer validation ✅

#2 - Bypass issuer validation ✅

#3 - Bypass network limitation

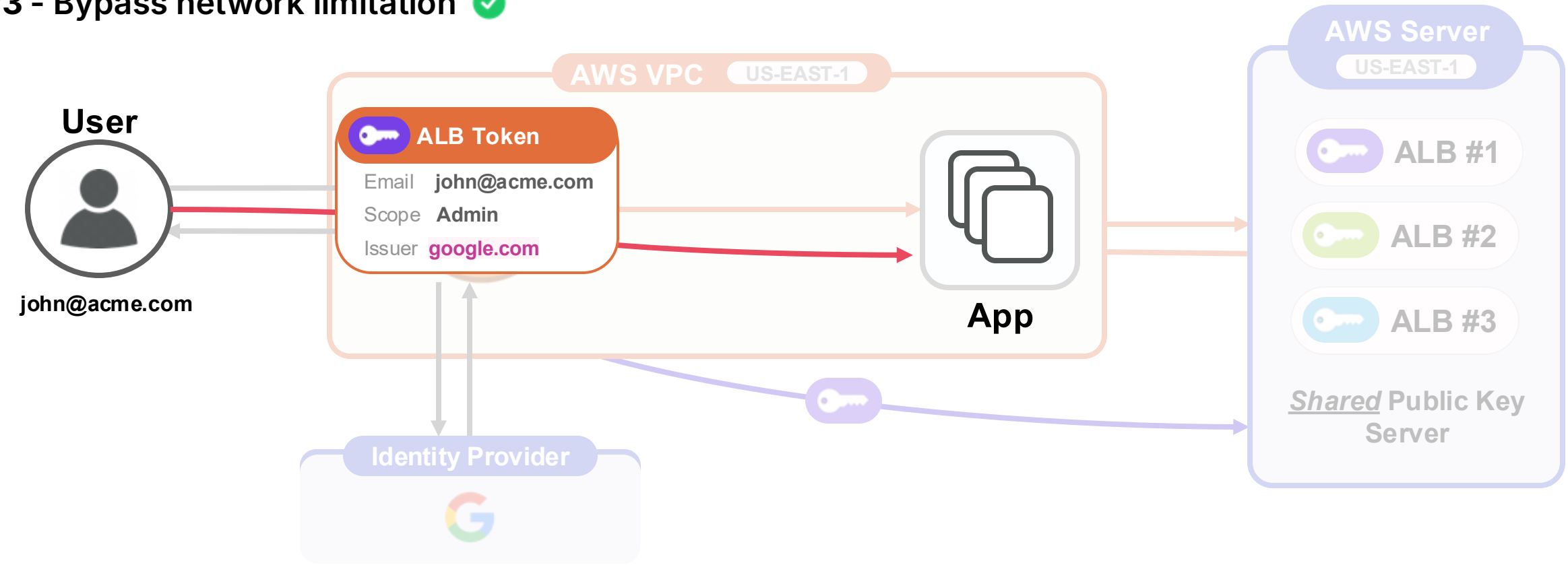


Three challenges to overcome

#1 - Bypass signer validation ✅

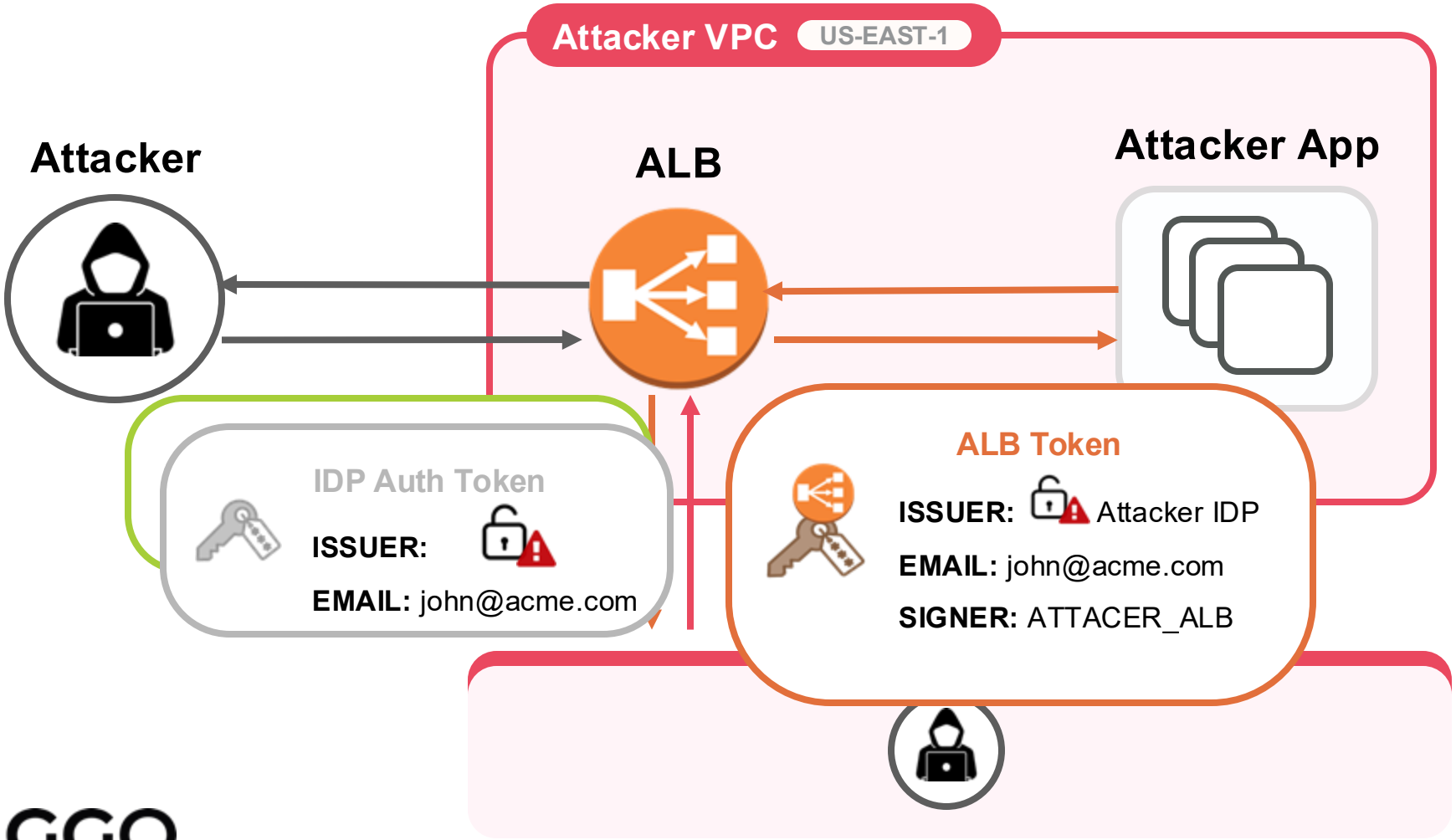
#2 - Bypass issuer validation ✅

#3 - Bypass network limitation ✅



ALBeast: Putting the parts together

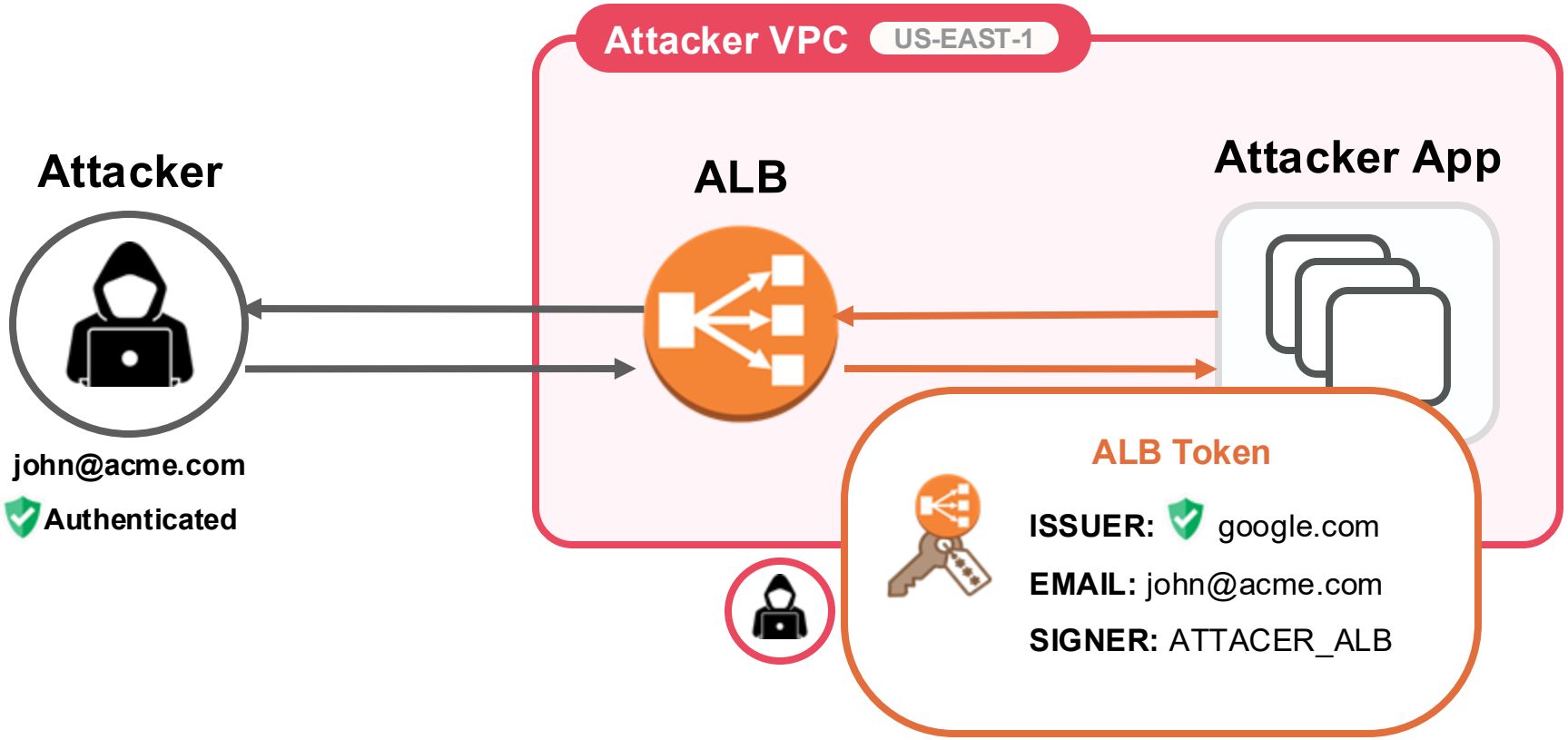
Attacker uses their own ALB in their environment



ALBeast: Putting the parts together

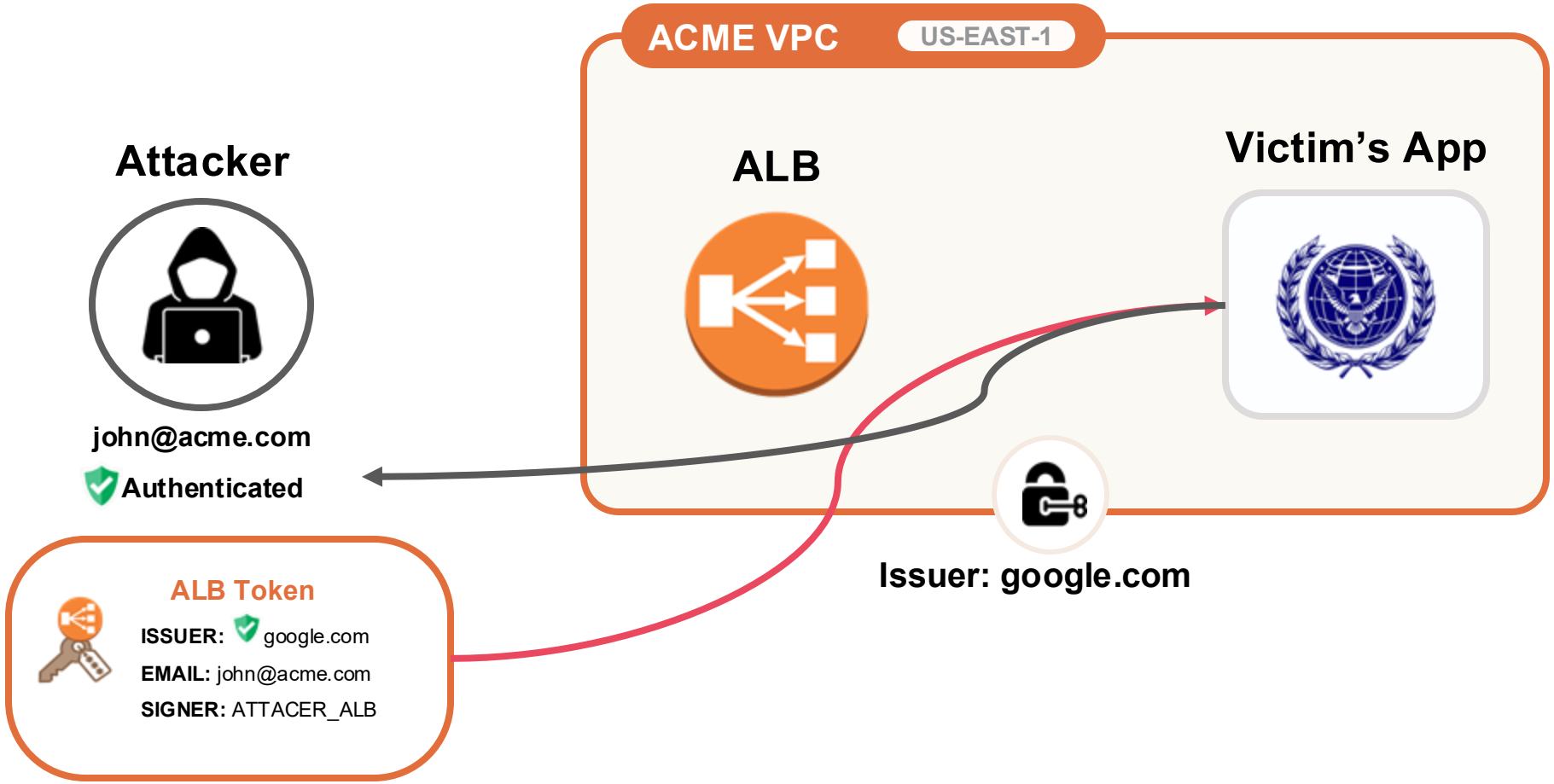
Attacker edits the issuer in the ALB config

ALB signs the token and sends it to the application



ALBeast: Putting the parts together

Attack Scenario



What's the impact?

Enumerating ALBs

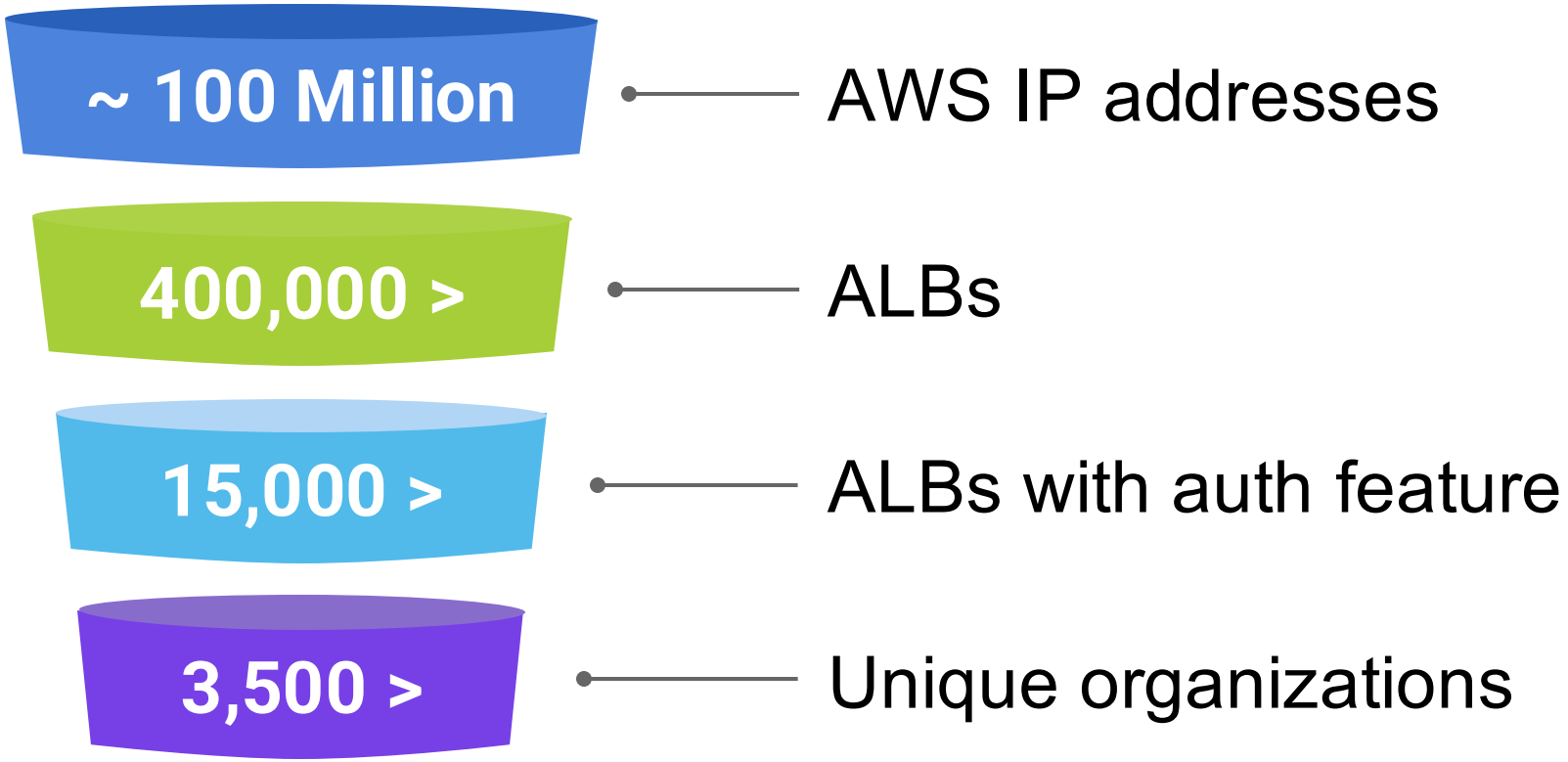
User

GET /oauth2/idpresponse

ALB

401 Authorization Required

What's the impact?



Disclosing the findings to AWS Security

- Reported to AWS on April 2024
- Initial response was a documentation update

```
import jwt

# Step 1: Validate the signer

expected_alb_arn = 'arn:aws:elasticloadbalancing:region-code:account-
id:loadbalancer/app/load-balancer-name/load-balancer-id'
```

… you must verify the signature before doing any authorization based on the claims and validate that the `signer` field in the JWT header contains the expected Application Load Balancer ARN.

Documentation change - is it enough?

- Customers are now required to:
  - Update application's code

> *"As a security best practice we recommend you restrict your targets to only receive traffic from your Application Load Balancer."*

  - Update to security group configurations

AWS' next actions

- AWS **contacts customers** with vulnerable configurations (**June 24'**)

ELB security notification

We are contacting you to ensure you properly configured your Application Load Balancer (ALB) and do not have target instances open to the public internet.

Please review your current configuration and verify your target security groups only allow traffic from the security group of the ALB by reference [1]. When using ALB's field matches the ARN of your ALB by referring to the code example in the section that states "...how to get the key ID, public key, and payload in Python 3.x" [2].

- AWS **adds a new security check** for ALB misconfigs (**November 24'**)

Application Load Balancer security group

Description

Check ID

`8604e947f2`

ALBeast makes waves

Forbes

INNOVATION > CYBERSECURITY

Unpacking The ALBeast Flaw

By Tony Bradley, Senior Contributor. Tony Bradley covers the intersection of tech and...

Follow Author

Aug 21, 2024, 08:18am EDT

Save Article Comment 0

ALBeast, a flaw within AWS Application Load ... [+]
GETTY

SECURITYWEEK

CLOUD SECURITY

Thousands of Apps Using AWS ALB Exposed to Attacks Due to Configuration Issue

As many as 15,000 applications using AWS Application Load Balancer (ALB) could be exposed to ALBeast attacks.

By Eduard Kovacs
August 21, 2024

As many as 15,000 apps that use AWS's Application Load Balancer (ALB) for authentication could be vulnerable to attacks, according to application security company Miggo.

These attacks, dubbed ALBeast by Miggo, are possible due to what the company has described as a critical configuration issue, rather than an actual vulnerability in the AWS ALB solution.

WIRED

LILY HAY NEWMAN

SECURITY AUG 20, 2024 6:00 PM

An AWS Configuration Issue Could Expose Thousands of Web Apps

Amazon has updated its instructions for how customers should more securely implement AWS's traffic-routing service known as Application Load Balancer, but it's not clear everyone will get the memo.

ALBeast is public

CVE-2024-10125 aws

CVE-2024-8901 aws

CVE-2025-1293 HashiCorp

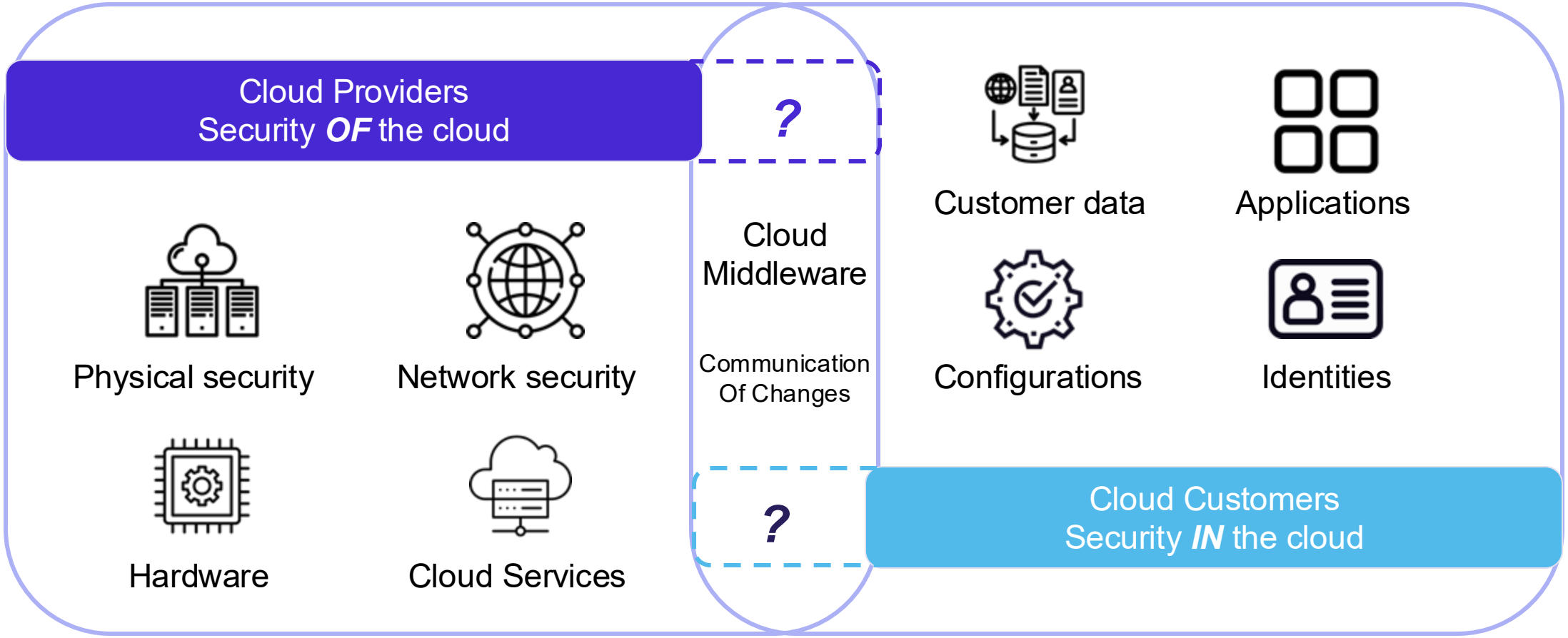
CVE-2025-25182 GCHQ

More under disclosure or private

RSAC 2025 Conference

ALBeast: Lessons Learned

The Cloud Shared Responsibility Model: 'It's Complicated'

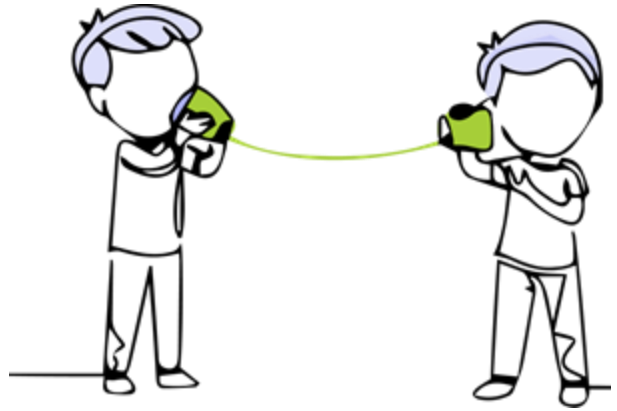


Incentives for Communication

- Customers can’t fix problems they’re unaware of

  Patch misconfigured instance ≠ Resolving the root cause

- Perverse incentive structure
  - Documentation updates aren’t always clearly communicated
  - The community should commend transparency



Communicating Well

- Transparency ≠ Nagging
- Our suggested criteria: action on the client's part
- Cloud-based Applications require more than CVE tracking

Was ALBeast a unique case?

- “Verified Access” feature: same architecture, same risks?

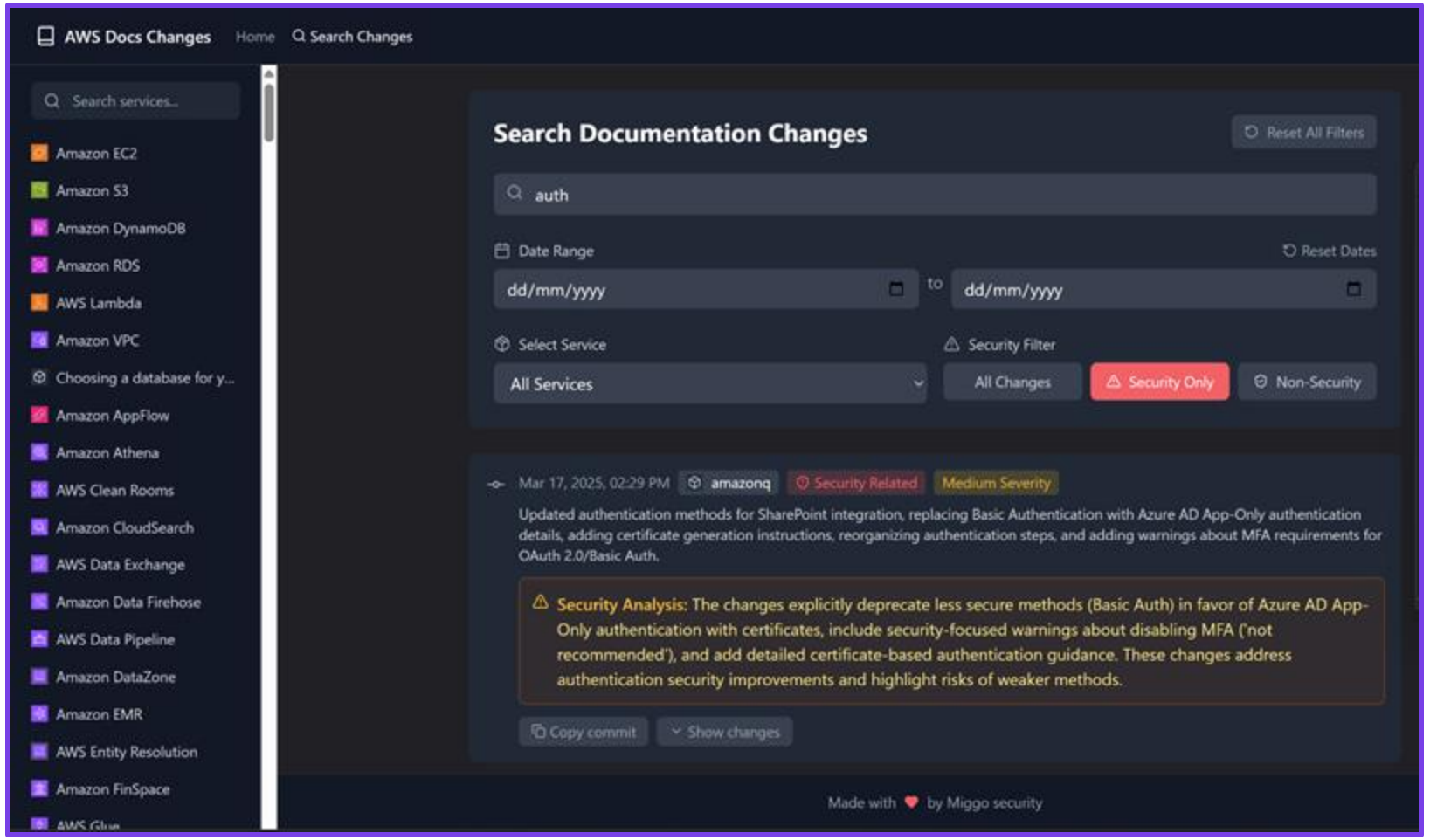
The endpoint for each AWS Region is as follows:

```
https://public-keys.prod.verified-access.<region>.amazonaws.com/<key-id>
```

- Same documentation update:

```
# Step 1: Validate the signer
expected_verified_access_instance_arn
```

AwsSecurityChanges.com



Applying what you learned today

- **Cloud Providers:**
  - Proactive updates on security related documentation changes
- **Security Teams:**
  - Implement AWS ALB Authentication best practices
  - Audit areas where the authentication logic is separate from the application
  - Track cloud architecture design issues continuously
  - Adopt runtime monitoring

RSAC 2025 Conference

Q&A

Thank you!

Liad Eliyahu, Head of Research @ Miggo Security

@liadeliyahu | liadeliyahu | liad@miggo.io

Itai Goldman, CTO @ Miggo Security

@itaig | itaigoldman | itai@miggo.io